

# iLab Neuromorphic Vision Toolkit Reference Manual

## Version 3.1

Generated by Doxygen 1.3.6

Mon Nov 8 04:18:33 2004



# Contents

|          |                                                                                 |           |
|----------|---------------------------------------------------------------------------------|-----------|
| <b>1</b> | <b><a href="#">iLab Neuromorphic Vision Toolkit Main Page</a></b>               | <b>1</b>  |
| <b>2</b> | <b><a href="#">iLab Neuromorphic Vision Toolkit Module Index</a></b>            | <b>3</b>  |
| 2.1      | <a href="#">iLab Neuromorphic Vision Toolkit Modules . . . . .</a>              | 3         |
| <b>3</b> | <b><a href="#">iLab Neuromorphic Vision Toolkit Namespace Index</a></b>         | <b>5</b>  |
| 3.1      | <a href="#">iLab Neuromorphic Vision Toolkit Namespace List . . . . .</a>       | 5         |
| <b>4</b> | <b><a href="#">iLab Neuromorphic Vision Toolkit Hierarchical Index</a></b>      | <b>7</b>  |
| 4.1      | <a href="#">iLab Neuromorphic Vision Toolkit Class Hierarchy . . . . .</a>      | 7         |
| <b>5</b> | <b><a href="#">iLab Neuromorphic Vision Toolkit Class Index</a></b>             | <b>21</b> |
| 5.1      | <a href="#">iLab Neuromorphic Vision Toolkit Class List . . . . .</a>           | 21        |
| <b>6</b> | <b><a href="#">iLab Neuromorphic Vision Toolkit File Index</a></b>              | <b>33</b> |
| 6.1      | <a href="#">iLab Neuromorphic Vision Toolkit File List . . . . .</a>            | 33        |
| <b>7</b> | <b><a href="#">iLab Neuromorphic Vision Toolkit Page Index</a></b>              | <b>49</b> |
| 7.1      | <a href="#">iLab Neuromorphic Vision Toolkit Related Pages . . . . .</a>        | 49        |
| <b>8</b> | <b><a href="#">iLab Neuromorphic Vision Toolkit Module Documentation</a></b>    | <b>51</b> |
| 8.1      | <a href="#">max-normalization algorithms . . . . .</a>                          | 51        |
| 8.2      | <a href="#">String conversion helpers . . . . .</a>                             | 61        |
| <b>9</b> | <b><a href="#">iLab Neuromorphic Vision Toolkit Namespace Documentation</a></b> | <b>75</b> |
| 9.1      | <a href="#">Png Namespace Reference . . . . .</a>                               | 75        |

|                                                                      |           |
|----------------------------------------------------------------------|-----------|
| 9.2 Raster Namespace Reference . . . . .                             | 77        |
| 9.3 std Namespace Reference . . . . .                                | 84        |
| <b>10 iLab Neuromorphic Vision Toolkit Class Documentation</b>       | <b>85</b> |
| 10.1 AppContext Class Reference . . . . .                            | 85        |
| 10.2 ArrayData< T > Class Template Reference . . . . .               | 87        |
| 10.3 ArrayHandle< T > Class Template Reference . . . . .             | 93        |
| 10.4 ArrayHelper< T, isTrivial > Struct Template Reference . . . . . | 97        |
| 10.5 ArrayHelper< T, true > Struct Template Reference . . . . .      | 100       |
| 10.6 AttentionGuidanceMap Class Reference . . . . .                  | 102       |
| 10.7 AttentionGuidanceMapConfigurator Class Reference . . . . .      | 108       |
| 10.8 AttentionGuidanceMapStd Class Reference . . . . .               | 112       |
| 10.9 AudioGrabber Class Reference . . . . .                          | 116       |
| 10.10AudioMixer Class Reference . . . . .                            | 121       |
| 10.11Beobot Class Reference . . . . .                                | 125       |
| 10.12BeobotAction Class Reference . . . . .                          | 130       |
| 10.13BeobotEffectors Class Reference . . . . .                       | 134       |
| 10.14BeobotMemory Class Reference . . . . .                          | 137       |
| 10.15BeobotSensation Class Reference . . . . .                       | 139       |
| 10.16BeobotSensors Class Reference . . . . .                         | 143       |
| 10.17BeobotVisualCortex Class Reference . . . . .                    | 144       |
| 10.18BeoChip Class Reference . . . . .                               | 151       |
| 10.19BeoChipListener Class Reference . . . . .                       | 169       |
| 10.20BeoSub Class Reference . . . . .                                | 171       |
| 10.21BeoSubAction Class Reference . . . . .                          | 181       |
| 10.22BeoSubActionDive Class Reference . . . . .                      | 187       |
| 10.23BeoSubActionTurn Class Reference . . . . .                      | 190       |
| 10.24BeoSubDepthSensor Class Reference . . . . .                     | 193       |
| 10.25BeoSubMotor Class Reference . . . . .                           | 196       |
| 10.26Beowulf Class Reference . . . . .                               | 200       |
| 10.27BestEntry Class Reference . . . . .                             | 208       |



|                                                              |     |
|--------------------------------------------------------------|-----|
| 10.28BitObject Class Reference . . . . .                     | 211 |
| 10.29BlueChannel Class Reference . . . . .                   | 222 |
| 10.30BlueChannelSurprise Class Reference . . . . .           | 224 |
| 10.31BlueYellowChannel Class Reference . . . . .             | 227 |
| 10.32BlueYellowChannelBeo Class Reference . . . . .          | 229 |
| 10.33BlueYellowChannelSurprise Class Reference . . . . .     | 231 |
| 10.34BlurFoveator Class Reference . . . . .                  | 234 |
| 10.35BPneuron Class Reference . . . . .                      | 237 |
| 10.36BPnnet Class Reference . . . . .                        | 240 |
| 10.37Brain Class Reference . . . . .                         | 243 |
| 10.38bug Class Reference . . . . .                           | 263 |
| 10.39cameraConf Class Reference . . . . .                    | 264 |
| 10.40CameraControl Class Reference . . . . .                 | 273 |
| 10.41carConf Class Reference . . . . .                       | 278 |
| 10.42CarControl Class Reference . . . . .                    | 283 |
| 10.43cascadeHold Class Reference . . . . .                   | 290 |
| 10.44ChannelBase Class Reference . . . . .                   | 293 |
| 10.45ChannelListener Class Reference . . . . .               | 310 |
| 10.46CheckedIterator< T > Class Template Reference . . . . . | 311 |
| 10.47CINNIC Class Reference . . . . .                        | 317 |
| 10.48CloseButtonListener Class Reference . . . . .           | 321 |
| 10.49ColorBandChannel Class Reference . . . . .              | 324 |
| 10.50ColorBandChannelBeo Class Reference . . . . .           | 326 |
| 10.51ColorBandChannelSurprise Class Reference . . . . .      | 328 |
| 10.52ColorChannel Class Reference . . . . .                  | 331 |
| 10.53ColorChannelBeo Class Reference . . . . .               | 335 |
| 10.54ColorChannelSurprise Class Reference . . . . .          | 337 |
| 10.55ComplexChannel Class Reference . . . . .                | 340 |
| 10.56COMPort Class Reference . . . . .                       | 356 |
| 10.57CompositeColorChannel Class Reference . . . . .         | 362 |
| 10.58CompositeColorChannelSurprise Class Reference . . . . . | 365 |

|                                                                       |     |
|-----------------------------------------------------------------------|-----|
| 10.59ContourConnection Class Reference . . . . .                      | 367 |
| 10.60ContourLayer Class Reference . . . . .                           | 368 |
| 10.61ContourLayerDynamic Class Reference . . . . .                    | 369 |
| 10.62ContourLayerStatic Class Reference . . . . .                     | 370 |
| 10.63ContourNeuronCreate< TC2 > Class Template Reference . . . . .    | 371 |
| 10.64ContourNeuronProp< CH, X, Y > Class Template Reference . . . . . | 376 |
| 10.65contourRun Class Reference . . . . .                             | 382 |
| 10.66ConvolveChannel Class Reference . . . . .                        | 386 |
| 10.67ConvolveChannelSurprise Class Reference . . . . .                | 388 |
| 10.68ConvolvePyrBuilder< T > Class Template Reference . . . . .       | 391 |
| 10.69CpuTimer Class Reference . . . . .                               | 393 |
| 10.70std::deque< T, Alloc > Class Template Reference . . . . .        | 396 |
| 10.71Dims Class Reference . . . . .                                   | 397 |
| 10.72DirectFeedChannel Class Reference . . . . .                      | 402 |
| 10.73DirectionChannel Class Reference . . . . .                       | 415 |
| 10.74DirectionChannelBeo Class Reference . . . . .                    | 418 |
| 10.75DirectionChannelSurprise Class Reference . . . . .               | 420 |
| 10.76DirTest Class Reference . . . . .                                | 423 |
| 10.77DisparityChannel Class Reference . . . . .                       | 425 |
| 10.78DummyChannel Class Reference . . . . .                           | 430 |
| 10.79DummyChannelSurprise Class Reference . . . . .                   | 434 |
| 10.80Edge Class Reference . . . . .                                   | 437 |
| 10.81EndPointChannel Class Reference . . . . .                        | 441 |
| 10.82EndPointChannelSurprise Class Reference . . . . .                | 445 |
| 10.83Entity Class Reference . . . . .                                 | 448 |
| 10.84Entity_TWM Class Reference . . . . .                             | 453 |
| 10.85EntropyChannel Class Reference . . . . .                         | 457 |
| 10.86EyeTrackerSaccadeController Class Reference . . . . .            | 461 |
| 10.87FeatureVector Class Reference . . . . .                          | 467 |
| 10.88FeedForwardNetwork Class Reference . . . . .                     | 470 |
| 10.89FixedSaccadeController Class Reference . . . . .                 | 474 |

|                                                                 |     |
|-----------------------------------------------------------------|-----|
| 10.90FlickerChannel Class Reference . . . . .                   | 479 |
| 10.91FlickerChannelBeo Class Reference . . . . .                | 483 |
| 10.92FlickerChannelSurprise Class Reference . . . . .           | 486 |
| 10.93FOEstimator Class Reference . . . . .                      | 489 |
| 10.94Foveator Class Reference . . . . .                         | 491 |
| 10.95FrameGrabber Class Reference . . . . .                     | 496 |
| 10.96FrameGrabberConfigurator Class Reference . . . . .         | 505 |
| 10.97FrameRange Class Reference . . . . .                       | 509 |
| 10.98FrameSeries Class Reference . . . . .                      | 512 |
| 10.99FrictionSaccadeController Class Reference . . . . .        | 523 |
| 10.10GaborChannel Class Reference . . . . .                     | 529 |
| 10.10GaborChannelBeo Class Reference . . . . .                  | 533 |
| 10.10GaborChannelSurprise Class Reference . . . . .             | 536 |
| 10.10GaborPyrBuilder< T > Class Template Reference . . . . .    | 539 |
| 10.10GACHromosome Class Reference . . . . .                     | 543 |
| 10.10GAPopulation Class Reference . . . . .                     | 550 |
| 10.10GaussianPyrBuilder< T > Class Template Reference . . . . . | 555 |
| 10.10GenericPyrBuilder< T > Class Template Reference . . . . .  | 557 |
| 10.10GetSaliency Class Reference . . . . .                      | 559 |
| 10.10GistEstimator Class Reference . . . . .                    | 563 |
| 10.11GistEstimatorConfigurator Class Reference . . . . .        | 566 |
| 10.11GpsDev Class Reference . . . . .                           | 570 |
| 10.11GreenChannel Class Reference . . . . .                     | 576 |
| 10.11GreenChannelSurprise Class Reference . . . . .             | 578 |
| 10.11Heading Class Reference . . . . .                          | 581 |
| 10.11Histogram Class Reference . . . . .                        | 584 |
| 10.11Hmax Class Reference . . . . .                             | 587 |
| 10.11HMR3300 Class Reference . . . . .                          | 593 |
| 10.11HIREntry Class Reference . . . . .                         | 596 |
| 10.11HueChannel Class Reference . . . . .                       | 599 |
| 10.12HyperRectangle Class Reference . . . . .                   | 604 |

|                                                                  |     |
|------------------------------------------------------------------|-----|
| 10.12IEEE1394grabber Class Reference . . . . .                   | 607 |
| 10.12Image< T > Class Template Reference . . . . .               | 611 |
| 10.12ImageCache< T > Class Template Reference . . . . .          | 657 |
| 10.12ImageCacheAvg< T > Class Template Reference . . . . .       | 664 |
| 10.12ImageCacheMinMax< T > Class Template Reference . . . . .    | 668 |
| 10.12ImageDisplay Class Reference . . . . .                      | 671 |
| 10.12ImageMap Class Reference . . . . .                          | 676 |
| 10.12ImageSet< T > Class Template Reference . . . . .            | 677 |
| 10.12ImageSpring< T > Class Template Reference . . . . .         | 684 |
| 10.13InferoTemporal Class Reference . . . . .                    | 691 |
| 10.13InferoTemporalConfigurator Class Reference . . . . .        | 695 |
| 10.13InformationChannel Class Reference . . . . .                | 699 |
| 10.13InputFrameSeries Class Reference . . . . .                  | 703 |
| 10.13IntensityChannel Class Reference . . . . .                  | 708 |
| 10.13IntensityChannelBeo Class Reference . . . . .               | 710 |
| 10.13IntensityChannelSurprise Class Reference . . . . .          | 713 |
| 10.13Jet< T > Class Template Reference . . . . .                 | 716 |
| 10.13JetSpec Class Reference . . . . .                           | 723 |
| 10.13JunctionChannel Class Reference . . . . .                   | 729 |
| 10.14JunctionChannelSurprise Class Reference . . . . .           | 733 |
| 10.14KalmanFilter Class Reference . . . . .                      | 736 |
| 10.14KDTree Class Reference . . . . .                            | 741 |
| 10.14KeyBoard Class Reference . . . . .                          | 744 |
| 10.14KeypointMatch Class Reference . . . . .                     | 746 |
| 10.14KnowledgeBase Class Reference . . . . .                     | 749 |
| 10.14LaplacianPyrBuilder< T > Class Template Reference . . . . . | 753 |
| 10.14Led Class Reference . . . . .                               | 755 |
| 10.14LeakyIntegrator Class Reference . . . . .                   | 760 |
| 10.14LeakyIntFire Class Reference . . . . .                      | 763 |
| 10.15LevelSpec Class Reference . . . . .                         | 766 |
| 10.15std::list< T, Alloc > Class Template Reference . . . . .    | 773 |

|                                                                                |     |
|--------------------------------------------------------------------------------|-----|
| 10.152JunctionChannel Class Reference . . . . .                                | 774 |
| 10.153JunctionChannelSurprise Class Reference . . . . .                        | 778 |
| 10.154LPTFoveator Class Reference . . . . .                                    | 781 |
| 10.155Map Class Reference . . . . .                                            | 787 |
| 10.156std::map< Key, Data, Compare, Alloc > Class Template Reference . . . . . | 791 |
| 10.157Matrix3 Class Reference . . . . .                                        | 792 |
| 10.158MbariResultViewer Class Reference . . . . .                              | 796 |
| 10.159Master::MetaData Class Reference . . . . .                               | 812 |
| 10.160MexBuf Class Reference . . . . .                                         | 817 |
| 10.161MexBufInit Class Reference . . . . .                                     | 818 |
| 10.162MexModelManager Class Reference . . . . .                                | 819 |
| 10.163MexReturn Class Reference . . . . .                                      | 824 |
| 10.164ModelComponent Class Reference . . . . .                                 | 827 |
| 10.165ModelEyeHeadController Class Reference . . . . .                         | 848 |
| 10.166ModelManager Class Reference . . . . .                                   | 854 |
| 10.167ModelOptionDef Struct Reference . . . . .                                | 868 |
| 10.168ModelParam< T > Class Template Reference . . . . .                       | 871 |
| 10.169ModelParamBase Class Reference . . . . .                                 | 875 |
| 10.170ModelWrapperSaccadeController Class Reference . . . . .                  | 881 |
| 10.171Monkey2SaccadeController Class Reference . . . . .                       | 886 |
| 10.172MonkeyEyeHeadController Class Reference . . . . .                        | 891 |
| 10.173MonkeySaccadeController Class Reference . . . . .                        | 897 |
| 10.174MotionChannel Class Reference . . . . .                                  | 903 |
| 10.175MotionChannelBeo Class Reference . . . . .                               | 907 |
| 10.176MotionChannelSurprise Class Reference . . . . .                          | 909 |
| 10.177MotTest Class Reference . . . . .                                        | 912 |
| 10.178MPEGStream Class Reference . . . . .                                     | 914 |
| 10.179MultiColorBandChannel Class Reference . . . . .                          | 919 |
| 10.180MultiColorBandChannelBeo Class Reference . . . . .                       | 923 |
| 10.181MultiColorBandChannelSurprise Class Reference . . . . .                  | 925 |
| 10.182MultiConvolveChannel Class Reference . . . . .                           | 928 |

|                                                                           |      |
|---------------------------------------------------------------------------|------|
| 10.183MultiConvolveChannelSurprise Class Reference . . . . .              | 932  |
| 10.184MultiDirectFeedChannel Class Reference . . . . .                    | 934  |
| 10.185std::multimap< Key, Data, Compare, Alloc > Class Template Reference | 938  |
| 10.186std::multiset< Key, Compare, Alloc > Class Template Reference . . . | 939  |
| 10.187MyBeoChipListener Class Reference . . . . .                         | 940  |
| 10.188NewClass Class Reference . . . . .                                  | 942  |
| 10.189ObjectBase Class Reference . . . . .                                | 944  |
| 10.190OldFrameSeries Class Reference . . . . .                            | 947  |
| 10.191Omni< T > Class Template Reference . . . . .                        | 948  |
| 10.192Ontology Class Reference . . . . .                                  | 950  |
| 10.193OptionParser Class Reference . . . . .                              | 953  |
| 10.194OptionSet Class Reference . . . . .                                 | 958  |
| 10.195OrientationChannel Class Reference . . . . .                        | 962  |
| 10.196OrientationChannelBeo Class Reference . . . . .                     | 969  |
| 10.197OrientationChannelSurprise Class Reference . . . . .                | 971  |
| 10.198OrientedPyrBuilder< T > Class Template Reference . . . . .          | 974  |
| 10.199OutputFrameSeries Class Reference . . . . .                         | 976  |
| 10.200ParamMap Class Reference . . . . .                                  | 980  |
| 10.201ParamMap::Error Class Reference . . . . .                           | 991  |
| 10.202PbmParser Class Reference . . . . .                                 | 992  |
| 10.203PedestrianChannel Class Reference . . . . .                         | 996  |
| 10.204PedestrianChannelSurprise Class Reference . . . . .                 | 999  |
| 10.205PfmParser Class Reference . . . . .                                 | 1002 |
| 10.206PixRGB< T > Class Template Reference . . . . .                      | 1005 |
| 10.207PN03contrastChannel Class Reference . . . . .                       | 1018 |
| 10.208PngParser Class Reference . . . . .                                 | 1021 |
| 10.209PngWriter Class Reference . . . . .                                 | 1024 |
| 10.210Point< T, N > Class Template Reference . . . . .                    | 1025 |
| 10.211Point2D Class Reference . . . . .                                   | 1029 |
| 10.212Point2DT Class Reference . . . . .                                  | 1039 |
| 10.213promote_trait< T1, T2 > Struct Template Reference . . . . .         | 1042 |

|                                                                         |      |
|-------------------------------------------------------------------------|------|
| 10.21Property Class Reference . . . . .                                 | 1043 |
| 10.21PropertyVectorSet Class Reference . . . . .                        | 1046 |
| 10.21PropHold Class Reference . . . . .                                 | 1049 |
| 10.21PsychoDisplay Class Reference . . . . .                            | 1054 |
| 10.21PsychoTask Struct Reference . . . . .                              | 1068 |
| 10.21PvmMessage Class Reference . . . . .                               | 1070 |
| 10.22PyrBuilder< T > Class Template Reference . . . . .                 | 1076 |
| 10.22PyrFoveator Class Reference . . . . .                              | 1079 |
| 10.22RadioDecoder Class Reference . . . . .                             | 1083 |
| 10.22Range< T > Class Template Reference . . . . .                      | 1088 |
| 10.22ReadConfig Class Reference . . . . .                               | 1093 |
| 10.22ReadMatrix Class Reference . . . . .                               | 1101 |
| 10.22Rectangle Class Reference . . . . .                                | 1102 |
| 10.22RedChannel Class Reference . . . . .                               | 1108 |
| 10.22RedChannelSurprise Class Reference . . . . .                       | 1110 |
| 10.22RedGreenChannel Class Reference . . . . .                          | 1113 |
| 10.23RedGreenChannelBeo Class Reference . . . . .                       | 1115 |
| 10.23RedGreenChannelSurprise Class Reference . . . . .                  | 1118 |
| 10.23ReichardtPyrBuilder< T > Class Template Reference . . . . .        | 1121 |
| 10.23Retina Class Reference . . . . .                                   | 1124 |
| 10.23RetinaConfigurator Class Reference . . . . .                       | 1130 |
| 10.23RetinaStd Class Reference . . . . .                                | 1134 |
| 10.23RGBConvolveChannel Class Reference . . . . .                       | 1141 |
| 10.23RGBConvolvePyrBuilder< T > Class Template Reference . . . . .      | 1144 |
| 10.23SaccadeController Class Reference . . . . .                        | 1147 |
| 10.23SaccadeControllerConfigurator Class Reference . . . . .            | 1162 |
| 10.24SafeCopyHelper< T, isTrivial > Struct Template Reference . . . . . | 1166 |
| 10.24SafeCopyHelper< T, true > Struct Template Reference . . . . .      | 1167 |
| 10.24SaliencyMap Class Reference . . . . .                              | 1168 |
| 10.24SaliencyMapConfigurator Class Reference . . . . .                  | 1176 |
| 10.24SaliencyMapFast Class Reference . . . . .                          | 1179 |

|        |                                                                                |      |
|--------|--------------------------------------------------------------------------------|------|
| 10.245 | <a href="#">SaliencyMapStd Class Reference</a>                                 | 1186 |
| 10.246 | <a href="#">SaliencyMapTrivial Class Reference</a>                             | 1193 |
| 10.247 | <a href="#">ScaleSpace Class Reference</a>                                     | 1198 |
| 10.248 | <a href="#">SccorrChannel Class Reference</a>                                  | 1201 |
| 10.249 | <a href="#">Serial Class Reference</a>                                         | 1205 |
| 10.250 | <a href="#">std::set&lt; Key, Compare, Alloc &gt; Class Template Reference</a> | 1215 |
| 10.251 | <a href="#">ShapeEstimator Class Reference</a>                                 | 1216 |
| 10.252 | <a href="#">SharedPtr&lt; T &gt; Class Template Reference</a>                  | 1225 |
| 10.253 | <a href="#">SimpleEyeHeadSaccadeController Class Reference</a>                 | 1231 |
| 10.254 | <a href="#">SimulationViewer Class Reference</a>                               | 1235 |
| 10.255 | <a href="#">SimulationViewerCompress Class Reference</a>                       | 1242 |
| 10.256 | <a href="#">SimulationViewerConfigurator Class Reference</a>                   | 1258 |
| 10.257 | <a href="#">SimulationViewerEyeMvt Class Reference</a>                         | 1262 |
| 10.258 | <a href="#">SimulationViewerEyeMvt2 Class Reference</a>                        | 1272 |
| 10.259 | <a href="#">SimulationViewerStd Class Reference</a>                            | 1283 |
| 10.260 | <a href="#">SingleChannel Class Reference</a>                                  | 1297 |
| 10.261 | <a href="#">SingleChannelBeo Class Reference</a>                               | 1323 |
| 10.262 | <a href="#">SingleChannelBeoServer Class Reference</a>                         | 1327 |
| 10.263 | <a href="#">SingleChannelSurprise Class Reference</a>                          | 1331 |
| 10.264 | <a href="#">SkinHueChannel Class Reference</a>                                 | 1337 |
| 10.265 | <a href="#">std::slist&lt; T, Alloc &gt; Class Template Reference</a>          | 1339 |
| 10.266 | <a href="#">SOBlueYellowChannel Class Reference</a>                            | 1340 |
| 10.267 | <a href="#">SOBlueYellowChannelSurprise Class Reference</a>                    | 1342 |
| 10.268 | <a href="#">SockServ Class Reference</a>                                       | 1345 |
| 10.269 | <a href="#">SOColorChannel Class Reference</a>                                 | 1352 |
| 10.270 | <a href="#">SOColorChannelSurprise Class Reference</a>                         | 1359 |
| 10.271 | <a href="#">SOGreenRedChannel Class Reference</a>                              | 1364 |
| 10.272 | <a href="#">SOGreenRedChannelSurprise Class Reference</a>                      | 1366 |
| 10.273 | <a href="#">SORedGreenChannel Class Reference</a>                              | 1369 |
| 10.274 | <a href="#">SORedGreenChannelSurprise Class Reference</a>                      | 1371 |
| 10.275 | <a href="#">SortedLimitedList&lt; T &gt; Class Template Reference</a>          | 1374 |



|                                                                    |      |
|--------------------------------------------------------------------|------|
| 10.276oxChannel Class Reference . . . . .                          | 1378 |
| 10.278OYellowBlueChannel Class Reference . . . . .                 | 1384 |
| 10.278OYellowBlueChannelSurprise Class Reference . . . . .         | 1386 |
| 10.279SpeedSensor Class Reference . . . . .                        | 1389 |
| 10.286SC Class Reference . . . . .                                 | 1390 |
| 10.285staircase Class Reference . . . . .                          | 1396 |
| 10.285stdBrain Class Reference . . . . .                           | 1402 |
| 10.285StereoChannel Class Reference . . . . .                      | 1405 |
| 10.285straightLine2D Class Reference . . . . .                     | 1411 |
| 10.285std::string Class Reference . . . . .                        | 1416 |
| 10.286SurpriseImage< T > Class Template Reference . . . . .        | 1417 |
| 10.285SurpriseMap< T > Class Template Reference . . . . .          | 1422 |
| 10.288SurpriseModel Class Reference . . . . .                      | 1425 |
| 10.285SurpriseModelSG Class Reference . . . . .                    | 1431 |
| 10.296SurpriseModelSP Class Reference . . . . .                    | 1436 |
| 10.29TaskRelevanceMap Class Reference . . . . .                    | 1441 |
| 10.29TaskRelevanceMapConfigurator Class Reference . . . . .        | 1447 |
| 10.293TaskRelevanceMapKillN Class Reference . . . . .              | 1451 |
| 10.294TaskRelevanceMapKillStatic Class Reference . . . . .         | 1456 |
| 10.295TaskRelevanceMapStd Class Reference . . . . .                | 1462 |
| 10.296TcorrChannel Class Reference . . . . .                       | 1468 |
| 10.297TCPcliServ Class Reference . . . . .                         | 1472 |
| 10.298TCPcommunicator Class Reference . . . . .                    | 1476 |
| 10.299TCPmessage Class Reference . . . . .                         | 1485 |
| 10.300TemplateMatchChannel Class Reference . . . . .               | 1494 |
| 10.30TemplateMatchChannelSurprise Class Reference . . . . .        | 1497 |
| 10.30TemplateMatchPyrBuilder Class Reference . . . . .             | 1500 |
| 10.303TestSuite Class Reference . . . . .                          | 1502 |
| 10.304ThresholdFrictionSaccadeController Class Reference . . . . . | 1508 |
| 10.305ThresholdSaccadeController Class Reference . . . . .         | 1513 |
| 10.306Timer Class Reference . . . . .                              | 1519 |

|                                                                    |      |
|--------------------------------------------------------------------|------|
| 10.307JunctionChannel Class Reference . . . . .                    | 1523 |
| 10.308JunctionChannelSurprise Class Reference . . . . .            | 1527 |
| 10.309oken Class Reference . . . . .                               | 1530 |
| 10.310oolkitInit Class Reference . . . . .                         | 1535 |
| 10.311rivialSaccadeController Class Reference . . . . .            | 1537 |
| 10.312TWM Class Reference . . . . .                                | 1540 |
| 10.313ype_if< T, B > Struct Template Reference . . . . .           | 1544 |
| 10.314ype_with_N_bits< T, N > Struct Template Reference . . . . .  | 1545 |
| 10.315ypeTraits< T > Struct Template Reference . . . . .           | 1546 |
| 10.316ypeTraits< PixRGB< T > > Struct Template Reference . . . . . | 1547 |
| 10.3174Lgrabber Class Reference . . . . .                          | 1548 |
| 10.3184LgrabberLight Class Reference . . . . .                     | 1552 |
| 10.3194LPalette Struct Reference . . . . .                         | 1554 |
| 10.320arianceChannel Class Reference . . . . .                     | 1556 |
| 10.321VCC4 Class Reference . . . . .                               | 1559 |
| 10.322std::vector< T, Alloc > Class Template Reference . . . . .   | 1571 |
| 10.323Vector2D Class Reference . . . . .                           | 1572 |
| 10.324Vector3 Class Reference . . . . .                            | 1581 |
| 10.325isBufSaccadeController Class Reference . . . . .             | 1588 |
| 10.326isualBuffer Class Reference . . . . .                        | 1592 |
| 10.327isualCortex Class Reference . . . . .                        | 1601 |
| 10.328isualCortexBeo Class Reference . . . . .                     | 1618 |
| 10.329isualCortexConfigurator Class Reference . . . . .            | 1622 |
| 10.330isualCortexEyeMvt Class Reference . . . . .                  | 1626 |
| 10.331isualCortexSurprise Class Reference . . . . .                | 1633 |
| 10.332isualEvent Class Reference . . . . .                         | 1636 |
| 10.333isualEventSet Class Reference . . . . .                      | 1643 |
| 10.334isualObject Class Reference . . . . .                        | 1650 |
| 10.335isualObjectDB Class Reference . . . . .                      | 1657 |
| 10.336isualObjectMatch Class Reference . . . . .                   | 1661 |
| 10.337WinnerTakeAll Class Reference . . . . .                      | 1665 |

|                                                                        |                 |
|------------------------------------------------------------------------|-----------------|
| 10.33 <del>8</del> WinnerTakeAllConfigurator Class Reference . . . . . | 1672            |
| 10.33 <del>9</del> WinnerTakeAllDummy Class Reference . . . . .        | 1676            |
| 10.34 <del>0</del> WinnerTakeAllFast Class Reference . . . . .         | 1681            |
| 10.34 <del>1</del> WinnerTakeAllStd Class Reference . . . . .          | 1686            |
| 10.34 <del>2</del> WTAwinner Class Reference . . . . .                 | 1691            |
| 10.34 <del>3</del> XJunctionChannel Class Reference . . . . .          | 1696            |
| 10.34 <del>4</del> XJunctionChannelSurprise Class Reference . . . . .  | 1700            |
| 10.34 <del>5</del> XWindow Class Reference . . . . .                   | 1703            |
| 10.34 <del>6</del> XWinManaged Class Reference . . . . .               | 1708            |
| 10.34 <del>7</del> YellowChannel Class Reference . . . . .             | 1715            |
| 10.34 <del>8</del> YellowChannelSurprise Class Reference . . . . .     | 1717            |
| <br><b>11 iLab Neuromorphic Vision Toolkit File Documentation</b>      | <br><b>1721</b> |
| 11.1 ArrayData.H File Reference . . . . .                              | 1721            |
| 11.2 atomic.H File Reference . . . . .                                 | 1726            |
| 11.3 AttentionGuidanceMap.C File Reference . . . . .                   | 1728            |
| 11.4 AttentionGuidanceMap.H File Reference . . . . .                   | 1729            |
| 11.5 AudioGrabber.C File Reference . . . . .                           | 1730            |
| 11.6 AudioGrabber.H File Reference . . . . .                           | 1731            |
| 11.7 AudioMixer.C File Reference . . . . .                             | 1732            |
| 11.8 AudioMixer.H File Reference . . . . .                             | 1733            |
| 11.9 beobot-defs.H File Reference . . . . .                            | 1734            |
| 11.10beobot-master.C File Reference . . . . .                          | 1736            |
| 11.11beobot-slave.C File Reference . . . . .                           | 1740            |
| 11.12Beobot.C File Reference . . . . .                                 | 1743            |
| 11.13Beobot.H File Reference . . . . .                                 | 1744            |
| 11.14BeobotAction.H File Reference . . . . .                           | 1745            |
| 11.15BeobotEffectors.C File Reference . . . . .                        | 1746            |
| 11.16BeobotEffectors.H File Reference . . . . .                        | 1747            |
| 11.17BeobotMemory.C File Reference . . . . .                           | 1748            |
| 11.18BeobotMemory.H File Reference . . . . .                           | 1749            |

|                                                    |      |
|----------------------------------------------------|------|
| 11.19BeobotSensation.C File Reference . . . . .    | 1751 |
| 11.20BeobotSensation.H File Reference . . . . .    | 1752 |
| 11.21BeobotSensors.C File Reference . . . . .      | 1754 |
| 11.22BeobotSensors.H File Reference . . . . .      | 1755 |
| 11.23BeobotVisualCortex.C File Reference . . . . . | 1756 |
| 11.24BeobotVisualCortex.H File Reference . . . . . | 1757 |
| 11.25beochannel-server.C File Reference . . . . .  | 1759 |
| 11.26BeoChip.C File Reference . . . . .            | 1760 |
| 11.27BeoChip.H File Reference . . . . .            | 1761 |
| 11.28beograb-master.C File Reference . . . . .     | 1763 |
| 11.29beograb.C File Reference . . . . .            | 1765 |
| 11.30BeoSub-defs.H File Reference . . . . .        | 1767 |
| 11.31BeoSub.C File Reference . . . . .             | 1771 |
| 11.32BeoSub.H File Reference . . . . .             | 1773 |
| 11.33BeoSubAction.C File Reference . . . . .       | 1776 |
| 11.34BeoSubAction.H File Reference . . . . .       | 1777 |
| 11.35BeoSubDepthSensor.C File Reference . . . . .  | 1778 |
| 11.36BeoSubDepthSensor.H File Reference . . . . .  | 1779 |
| 11.37Beowulf.C File Reference . . . . .            | 1780 |
| 11.38Beowulf.H File Reference . . . . .            | 1781 |
| 11.39BitObject.C File Reference . . . . .          | 1784 |
| 11.40BitObject.H File Reference . . . . .          | 1786 |
| 11.41BitObjectDrawModes.H File Reference . . . . . | 1788 |
| 11.42blobProp.H File Reference . . . . .           | 1790 |
| 11.43BlurFoveator.C File Reference . . . . .       | 1791 |
| 11.44BlurFoveator.H File Reference . . . . .       | 1792 |
| 11.45bmcvfigs.C File Reference . . . . .           | 1794 |
| 11.46BPneuron.C File Reference . . . . .           | 1796 |
| 11.47BPneuron.H File Reference . . . . .           | 1797 |
| 11.48BPnnet.C File Reference . . . . .             | 1798 |
| 11.49BPnnet.H File Reference . . . . .             | 1799 |

|                                                    |      |
|----------------------------------------------------|------|
| 11.50Brain.C File Reference . . . . .              | 1801 |
| 11.51Brain.H File Reference . . . . .              | 1803 |
| 11.52calibrateCamera.C File Reference . . . . .    | 1806 |
| 11.53calibrateFunctions.C File Reference . . . . . | 1807 |
| 11.54calibrateFunctions.H File Reference . . . . . | 1808 |
| 11.55calibrateVCC4.C File Reference . . . . .      | 1809 |
| 11.56cameraConf.C File Reference . . . . .         | 1810 |
| 11.57cameraConf.H File Reference . . . . .         | 1811 |
| 11.58CameraControl.C File Reference . . . . .      | 1812 |
| 11.59CameraControl.H File Reference . . . . .      | 1813 |
| 11.60carConf.C File Reference . . . . .            | 1815 |
| 11.61carConf.H File Reference . . . . .            | 1816 |
| 11.62carcontrol.C File Reference . . . . .         | 1817 |
| 11.63carcontrol.H File Reference . . . . .         | 1818 |
| 11.64cascadeHold.H File Reference . . . . .        | 1820 |
| 11.65ChannelBase.C File Reference . . . . .        | 1821 |
| 11.66ChannelBase.H File Reference . . . . .        | 1822 |
| 11.67Channels.C File Reference . . . . .           | 1825 |
| 11.68Channels.H File Reference . . . . .           | 1828 |
| 11.69ChannelsBeo.C File Reference . . . . .        | 1834 |
| 11.70ChannelsBeo.H File Reference . . . . .        | 1835 |
| 11.71ChannelsSurprise.H File Reference . . . . .   | 1837 |
| 11.72CheckedIterator.H File Reference . . . . .    | 1840 |
| 11.73CINNIC.C File Reference . . . . .             | 1841 |
| 11.74CINNIC.H File Reference . . . . .             | 1842 |
| 11.75CINNICanova.C File Reference . . . . .        | 1844 |
| 11.76CINNICinteract.C File Reference . . . . .     | 1848 |
| 11.77CINNICstats.C File Reference . . . . .        | 1851 |
| 11.78CINNICstatsRun.C File Reference . . . . .     | 1854 |
| 11.79CINNICstatsRun.H File Reference . . . . .     | 1855 |
| 11.80CINNICtest.C File Reference . . . . .         | 1856 |

|                                                        |      |
|--------------------------------------------------------|------|
| 11.81ColorComputeTypes.H File Reference . . . . .      | 1859 |
| 11.82colorDefs.H File Reference . . . . .              | 1861 |
| 11.83ComplexChannel.C File Reference . . . . .         | 1862 |
| 11.84ComplexChannel.H File Reference . . . . .         | 1863 |
| 11.85COMPort.C File Reference . . . . .                | 1865 |
| 11.86COMPort.H File Reference . . . . .                | 1866 |
| 11.87computeClampedDiffMean.C File Reference . . . . . | 1868 |
| 11.88Contour.C File Reference . . . . .                | 1869 |
| 11.89contourDefine.H File Reference . . . . .          | 1872 |
| 11.90contourNeuron.C File Reference . . . . .          | 1873 |
| 11.91contourNeuron.H File Reference . . . . .          | 1875 |
| 11.92contourNeuronProp.C File Reference . . . . .      | 1877 |
| 11.93contourNeuronProp.H File Reference . . . . .      | 1878 |
| 11.94contourRun.C File Reference . . . . .             | 1879 |
| 11.95contourRun.H File Reference . . . . .             | 1880 |
| 11.96covEstimate.C File Reference . . . . .            | 1882 |
| 11.97covEstimate.H File Reference . . . . .            | 1883 |
| 11.98covHolder.H File Reference . . . . .              | 1884 |
| 11.99create_thumbnails.C File Reference . . . . .      | 1885 |
| 11.100Dims.H File Reference . . . . .                  | 1886 |
| 11.101DirCTest.C File Reference . . . . .              | 1888 |
| 11.102DirectFeedChannel.C File Reference . . . . .     | 1890 |
| 11.103DirectFeedChannel.H File Reference . . . . .     | 1891 |
| 11.104DirTest.C File Reference . . . . .               | 1893 |
| 11.105DirTest.H File Reference . . . . .               | 1895 |
| 11.106DummySTL.H File Reference . . . . .              | 1896 |
| 11.107eyemap.C File Reference . . . . .                | 1897 |
| 11.108gzvision.C File Reference . . . . .              | 1899 |
| 11.109hide_image.C File Reference . . . . .            | 1900 |
| 11.110fancynorm.C File Reference . . . . .             | 1901 |
| 11.111fancynorm.H File Reference . . . . .             | 1903 |

|                                                       |      |
|-------------------------------------------------------|------|
| 11.112CV_STM.C File Reference . . . . .               | 1908 |
| 11.113CV_STM.H File Reference . . . . .               | 1909 |
| 11.114FeatureClusterVision.C File Reference . . . . . | 1910 |
| 11.115FFN.C File Reference . . . . .                  | 1911 |
| 11.116FFN.H File Reference . . . . .                  | 1912 |
| 11.117FOEstimator.C File Reference . . . . .          | 1913 |
| 11.118FOEstimator.H File Reference . . . . .          | 1914 |
| 11.119Font2c.C File Reference . . . . .               | 1915 |
| 11.120Foveator.C File Reference . . . . .             | 1917 |
| 11.121Foveator.H File Reference . . . . .             | 1918 |
| 11.122Ipe.C File Reference . . . . .                  | 1919 |
| 11.123Ipe.H File Reference . . . . .                  | 1922 |
| 11.124FrameGrabber.C File Reference . . . . .         | 1925 |
| 11.125FrameGrabber.H File Reference . . . . .         | 1926 |
| 11.126FrameGrabberMode.H File Reference . . . . .     | 1928 |
| 11.127FrameRange.H File Reference . . . . .           | 1930 |
| 11.128FrameSeries.C File Reference . . . . .          | 1931 |
| 11.129FrameSeries.H File Reference . . . . .          | 1932 |
| 11.130Izvision.C File Reference . . . . .             | 1935 |
| 11.131LaborElement.C File Reference . . . . .         | 1936 |
| 11.132LaborElement.H File Reference . . . . .         | 1937 |
| 11.133LaborPyramid.C File Reference . . . . .         | 1938 |
| 11.134LaborPyramid.H File Reference . . . . .         | 1940 |
| 11.135LACHromosome.C File Reference . . . . .         | 1942 |
| 11.136LACHromosome.H File Reference . . . . .         | 1944 |
| 11.137LAMPiVision.C File Reference . . . . .          | 1946 |
| 11.138LAPopulation.C File Reference . . . . .         | 1948 |
| 11.139LAPopulation.H File Reference . . . . .         | 1949 |
| 11.140Geometry2D.C File Reference . . . . .           | 1951 |
| 11.141Geometry2D.H File Reference . . . . .           | 1953 |
| 11.142GetSaliency.C File Reference . . . . .          | 1955 |

|        |                                                      |      |
|--------|------------------------------------------------------|------|
| 11.143 | getSaliency.H File Reference . . . . .               | 1957 |
| 11.144 | getShapeEstimator.C File Reference . . . . .         | 1958 |
| 11.145 | GistEstimator.C File Reference . . . . .             | 1961 |
| 11.146 | GistEstimator.H File Reference . . . . .             | 1964 |
| 11.147 | GistEstimatorConfigurator.C File Reference . . . . . | 1965 |
| 11.148 | GistEstimatorConfigurator.H File Reference . . . . . | 1966 |
| 11.149 | GPS.C File Reference . . . . .                       | 1967 |
| 11.150 | GPS.H File Reference . . . . .                       | 1968 |
| 11.151 | grab-server.C File Reference . . . . .               | 1970 |
| 11.152 | Graph.C File Reference . . . . .                     | 1972 |
| 11.153 | Graph.H File Reference . . . . .                     | 1973 |
| 11.154 | gtk-SoxChannel.C File Reference . . . . .            | 1975 |
| 11.155 | Gyro.C File Reference . . . . .                      | 1979 |
| 11.156 | Gyro.H File Reference . . . . .                      | 1980 |
| 11.157 | Histogram.C File Reference . . . . .                 | 1981 |
| 11.158 | Histogram.H File Reference . . . . .                 | 1982 |
| 11.159 | Imax.C File Reference . . . . .                      | 1983 |
| 11.160 | Imax.H File Reference . . . . .                      | 1984 |
| 11.161 | HMR3300.C File Reference . . . . .                   | 1985 |
| 11.162 | HMR3300.H File Reference . . . . .                   | 1986 |
| 11.163 | IEEE1394grabber.C File Reference . . . . .           | 1988 |
| 11.164 | IEEE1394grabber.H File Reference . . . . .           | 1989 |
| 11.165 | Image.C File Reference . . . . .                     | 1992 |
| 11.166 | Image.H File Reference . . . . .                     | 1993 |
| 11.167 | Image_All.H File Reference . . . . .                 | 1994 |
| 11.168 | Image_ColorOps.C File Reference . . . . .            | 1997 |
| 11.169 | Image_ColorOps.H File Reference . . . . .            | 2016 |
| 11.170 | Image_FilterOps.C File Reference . . . . .           | 2035 |
| 11.171 | Image_FilterOps.H File Reference . . . . .           | 2049 |
| 11.172 | Image_IO.C File Reference . . . . .                  | 2069 |
| 11.173 | Image_IO.H File Reference . . . . .                  | 2072 |



|        |                                                       |      |
|--------|-------------------------------------------------------|------|
| 11.174 | Image_MathOps.C File Reference . . . . .              | 2075 |
| 11.175 | Image_MathOps.H File Reference . . . . .              | 2094 |
| 11.176 | Image_MonoOnly.C File Reference . . . . .             | 2116 |
| 11.177 | Image_Omni.C File Reference . . . . .                 | 2117 |
| 11.178 | Image_Omni.H File Reference . . . . .                 | 2121 |
| 11.179 | Image_ShapeOps.C File Reference . . . . .             | 2125 |
| 11.180 | Image_ShapeOps.H File Reference . . . . .             | 2128 |
| 11.181 | Image_Transforms.C File Reference . . . . .           | 2138 |
| 11.182 | Image_Transforms.H File Reference . . . . .           | 2147 |
| 11.183 | ImageCache.C File Reference . . . . .                 | 2158 |
| 11.184 | ImageCache.H File Reference . . . . .                 | 2160 |
| 11.185 | ImageConvertQt.C File Reference . . . . .             | 2162 |
| 11.186 | ImageConvertQt.H File Reference . . . . .             | 2164 |
| 11.187 | ImageDisplay.C File Reference . . . . .               | 2166 |
| 11.188 | ImageDisplay.H File Reference . . . . .               | 2167 |
| 11.189 | ImageDisplayTypes.H File Reference . . . . .          | 2168 |
| 11.190 | ImageMap.H File Reference . . . . .                   | 2170 |
| 11.191 | ImageSet.C File Reference . . . . .                   | 2171 |
| 11.192 | ImageSet.H File Reference . . . . .                   | 2172 |
| 11.193 | ImageSet_Ops.C File Reference . . . . .               | 2175 |
| 11.194 | ImageSet_Ops.H File Reference . . . . .               | 2187 |
| 11.195 | ImageSpring.C File Reference . . . . .                | 2200 |
| 11.196 | ImageSpring.H File Reference . . . . .                | 2201 |
| 11.197 | InferoTemporal.C File Reference . . . . .             | 2203 |
| 11.198 | InferoTemporal.H File Reference . . . . .             | 2204 |
| 11.199 | InferoTemporalConfigurator.C File Reference . . . . . | 2205 |
| 11.200 | InferoTemporalConfigurator.H File Reference . . . . . | 2206 |
| 11.201 | InstantiateAll.H File Reference . . . . .             | 2207 |
| 11.202 | ORtypes.H File Reference . . . . .                    | 2208 |
| 11.203 | Jet.C File Reference . . . . .                        | 2210 |
| 11.204 | Jet.H File Reference . . . . .                        | 2212 |

|                                                                |      |
|----------------------------------------------------------------|------|
| 11.20 <del>K</del> almanFilter.C File Reference . . . . .      | 2215 |
| 11.20 <del>K</del> almanFilter.H File Reference . . . . .      | 2216 |
| 11.20 <del>K</del> DTree.C File Reference . . . . .            | 2217 |
| 11.20 <del>K</del> DTree.H File Reference . . . . .            | 2218 |
| 11.20 <del>K</del> eyBoard.C File Reference . . . . .          | 2220 |
| 11.21 <del>K</del> eyBoard.H File Reference . . . . .          | 2221 |
| 11.21 <del>K</del> eypoint.C File Reference . . . . .          | 2223 |
| 11.21 <del>K</del> eypoint.H File Reference . . . . .          | 2224 |
| 11.21 <del>K</del> nowledgeBase.C File Reference . . . . .     | 2225 |
| 11.21 <del>K</del> nowledgeBase.H File Reference . . . . .     | 2226 |
| 11.21 <del>L</del> andmark_Graph.C File Reference . . . . .    | 2227 |
| 11.21 <del>L</del> andmark_Graph.H File Reference . . . . .    | 2228 |
| 11.21 <del>L</del> cd.C File Reference . . . . .               | 2230 |
| 11.21 <del>L</del> cd.H File Reference . . . . .               | 2231 |
| 11.21 <del>L</del> cdcho.C File Reference . . . . .            | 2232 |
| 11.22 <del>L</del> eakyIntegrator.H File Reference . . . . .   | 2233 |
| 11.22 <del>L</del> eakyIntFire.H File Reference . . . . .      | 2234 |
| 11.22 <del>L</del> earnvision.C File Reference . . . . .       | 2235 |
| 11.22 <del>L</del> evelSpec.C File Reference . . . . .         | 2237 |
| 11.22 <del>L</del> evelSpec.H File Reference . . . . .         | 2238 |
| 11.22 <del>L</del> og.C File Reference . . . . .               | 2241 |
| 11.22 <del>L</del> og.H File Reference . . . . .               | 2243 |
| 11.22 <del>L</del> PTFoveator.C File Reference . . . . .       | 2253 |
| 11.22 <del>L</del> PTFoveator.H File Reference . . . . .       | 2254 |
| 11.22 <del>L</del> umin.C File Reference . . . . .             | 2256 |
| 11.23 <del>M</del> ake_map.C File Reference . . . . .          | 2259 |
| 11.23 <del>M</del> ake_map.H File Reference . . . . .          | 2260 |
| 11.23 <del>M</del> ake_target_image.C File Reference . . . . . | 2261 |
| 11.23 <del>M</del> ake_test_movie.C File Reference . . . . .   | 2262 |
| 11.23 <del>M</del> bariFunctions.C File Reference . . . . .    | 2263 |
| 11.23 <del>M</del> bariFunctions.H File Reference . . . . .    | 2266 |

|        |                                                     |      |
|--------|-----------------------------------------------------|------|
| 11.236 | MbarResultViewer.C File Reference . . . . .         | 2269 |
| 11.237 | MbarResultViewer.H File Reference . . . . .         | 2271 |
| 11.238 | hexConverts.C File Reference . . . . .              | 2272 |
| 11.239 | hexConverts.H File Reference . . . . .              | 2278 |
| 11.240 | hexDummy.C File Reference . . . . .                 | 2284 |
| 11.241 | hexLog.C File Reference . . . . .                   | 2285 |
| 11.242 | hexLog.H File Reference . . . . .                   | 2287 |
| 11.243 | hexModelManager.C File Reference . . . . .          | 2290 |
| 11.244 | hexModelManager.H File Reference . . . . .          | 2291 |
| 11.245 | hexSaliency.C File Reference . . . . .              | 2292 |
| 11.246 | hexTest.C File Reference . . . . .                  | 2295 |
| 11.247 | hexTestDirectFeedChannel.C File Reference . . . . . | 2297 |
| 11.248 | hexTestReichardt.C File Reference . . . . .         | 2299 |
| 11.249 | hexVision.C File Reference . . . . .                | 2301 |
| 11.250 | mmx-sse.C File Reference . . . . .                  | 2304 |
| 11.251 | mmx-sse.H File Reference . . . . .                  | 2305 |
| 11.252 | ModelComponent.C File Reference . . . . .           | 2307 |
| 11.253 | ModelComponent.H File Reference . . . . .           | 2308 |
| 11.254 | ModelManager.C File Reference . . . . .             | 2311 |
| 11.255 | ModelManager.H File Reference . . . . .             | 2313 |
| 11.256 | ModelManagerControl.ui.h File Reference . . . . .   | 2316 |
| 11.257 | ModelManagerDialog.ui.h File Reference . . . . .    | 2317 |
| 11.258 | ModelOptionAliases.H File Reference . . . . .       | 2318 |
| 11.259 | ModelOptionDefs.C File Reference . . . . .          | 2319 |
| 11.260 | ModelOptionDefs.H File Reference . . . . .          | 2321 |
| 11.261 | ModelParam.C File Reference . . . . .               | 2329 |
| 11.262 | ModelParam.H File Reference . . . . .               | 2332 |
| 11.263 | MotCTest.C File Reference . . . . .                 | 2333 |
| 11.264 | MotTest.C File Reference . . . . .                  | 2335 |
| 11.265 | MotTest.H File Reference . . . . .                  | 2337 |
| 11.266 | MPEGStream.C File Reference . . . . .               | 2339 |

|       |                                                |      |
|-------|------------------------------------------------|------|
| 11.26 | MPEGStream.H File Reference . . . . .          | 2343 |
| 11.26 | mpgsize.C File Reference . . . . .             | 2345 |
| 11.26 | MPIopenvision.C File Reference . . . . .       | 2346 |
| 11.27 | MPIopenvision2.C File Reference . . . . .      | 2348 |
| 11.27 | MPIopenvision3.C File Reference . . . . .      | 2351 |
| 11.27 | MrGabor.C File Reference . . . . .             | 2354 |
| 11.27 | MrGabor2.C File Reference . . . . .            | 2359 |
| 11.27 | herd-cam-wan.C File Reference . . . . .        | 2364 |
| 11.27 | herd-camTCP2-master.C File Reference . . . . . | 2366 |
| 11.27 | herd-camTCP2.C File Reference . . . . .        | 2371 |
| 11.27 | NewClass.C File Reference . . . . .            | 2374 |
| 11.27 | NewClass.H File Reference . . . . .            | 2375 |
| 11.27 | NPclassify-test.C File Reference . . . . .     | 2376 |
| 11.28 | NPclassify.C File Reference . . . . .          | 2378 |
| 11.28 | NPclassify.H File Reference . . . . .          | 2379 |
| 11.28 | NPclassify2.C File Reference . . . . .         | 2380 |
| 11.28 | NPclassify2.H File Reference . . . . .         | 2381 |
| 11.28 | Omni.C File Reference . . . . .                | 2382 |
| 11.28 | Omni.H File Reference . . . . .                | 2385 |
| 11.28 | openvision.C File Reference . . . . .          | 2386 |
| 11.28 | openvision3.C File Reference . . . . .         | 2388 |
| 11.28 | OptionParser.C File Reference . . . . .        | 2390 |
| 11.28 | OptionParser.H File Reference . . . . .        | 2391 |
| 11.29 | OrientComputeTypes.H File Reference . . . . .  | 2392 |
| 11.29 | ParamMap.C File Reference . . . . .            | 2394 |
| 11.29 | ParamMap.H File Reference . . . . .            | 2395 |
| 11.29 | PbmParser.C File Reference . . . . .           | 2396 |
| 11.29 | PbmParser.H File Reference . . . . .           | 2397 |
| 11.29 | PfmParser.C File Reference . . . . .           | 2398 |
| 11.29 | PfmParser.H File Reference . . . . .           | 2399 |
| 11.29 | Pixels.H File Reference . . . . .              | 2400 |

|                                                      |      |
|------------------------------------------------------|------|
| 11.29PngParser.C File Reference . . . . .            | 2404 |
| 11.29PngParser.H File Reference . . . . .            | 2405 |
| 11.30PngWriter.C File Reference . . . . .            | 2406 |
| 11.30PngWriter.H File Reference . . . . .            | 2407 |
| 11.30Point.H File Reference . . . . .                | 2408 |
| 11.30Point2D.H File Reference . . . . .              | 2409 |
| 11.30Point2DT.H File Reference . . . . .             | 2412 |
| 11.30PointAndFlood.C File Reference . . . . .        | 2413 |
| 11.30PolatSagiTest.C File Reference . . . . .        | 2416 |
| 11.30Prepare_target_image.C File Reference . . . . . | 2421 |
| 11.30Promotions.H File Reference . . . . .           | 2422 |
| 11.30PSNR.C File Reference . . . . .                 | 2424 |
| 11.31Psycho-dsearch.C File Reference . . . . .       | 2425 |
| 11.31Psycho-headmove.C File Reference . . . . .      | 2427 |
| 11.31Psycho-movie.C File Reference . . . . .         | 2428 |
| 11.31Psycho-movie2.C File Reference . . . . .        | 2429 |
| 11.31Psycho-obfba.C File Reference . . . . .         | 2431 |
| 11.31Psycho-search.C File Reference . . . . .        | 2434 |
| 11.31Psycho-still.C File Reference . . . . .         | 2436 |
| 11.31PsychoDisplay.C File Reference . . . . .        | 2437 |
| 11.31PsychoDisplay.H File Reference . . . . .        | 2439 |
| 11.31Pvision-master.C File Reference . . . . .       | 2441 |
| 11.32Pvision.C File Reference . . . . .              | 2443 |
| 11.32PvisionTCP-defs.H File Reference . . . . .      | 2445 |
| 11.32PvisionTCP-master.C File Reference . . . . .    | 2447 |
| 11.32PvisionTCP.C File Reference . . . . .           | 2450 |
| 11.32PvisionTCP2-master.C File Reference . . . . .   | 2454 |
| 11.32PvisionTCP2.C File Reference . . . . .          | 2458 |
| 11.32PvisionTCP3-master.C File Reference . . . . .   | 2462 |
| 11.32PvisionTCP3.C File Reference . . . . .          | 2466 |
| 11.32Pvmdefs.H File Reference . . . . .              | 2470 |

|                                                    |      |
|----------------------------------------------------|------|
| 11.329vmMessage.C File Reference . . . . .         | 2471 |
| 11.330vmMessage.H File Reference . . . . .         | 2472 |
| 11.33Pyramid_Ops.C File Reference . . . . .        | 2474 |
| 11.33Pyramid_Ops.H File Reference . . . . .        | 2480 |
| 11.33PyramidTypes.H File Reference . . . . .       | 2489 |
| 11.33PyrBuilder.C File Reference . . . . .         | 2491 |
| 11.33PyrBuilder.H File Reference . . . . .         | 2492 |
| 11.33PyrFoveator.C File Reference . . . . .        | 2494 |
| 11.33PyrFoveator.H File Reference . . . . .        | 2495 |
| 11.33RadioDecoder.C File Reference . . . . .       | 2497 |
| 11.33RadioDecoder.H File Reference . . . . .       | 2499 |
| 11.34Range.H File Reference . . . . .              | 2500 |
| 11.34Raster.C File Reference . . . . .             | 2501 |
| 11.34Raster.H File Reference . . . . .             | 2504 |
| 11.34RasterFileFormat.H File Reference . . . . .   | 2505 |
| 11.34ReadConfig.C File Reference . . . . .         | 2506 |
| 11.34ReadConfig.H File Reference . . . . .         | 2507 |
| 11.34Rectangle.H File Reference . . . . .          | 2508 |
| 11.34Retina.C File Reference . . . . .             | 2509 |
| 11.34Retina.C File Reference . . . . .             | 2511 |
| 11.34Retina.H File Reference . . . . .             | 2512 |
| 11.35SaccadeController.C File Reference . . . . .  | 2513 |
| 11.35SaccadeController.H File Reference . . . . .  | 2514 |
| 11.35SaccadeControllers.C File Reference . . . . . | 2515 |
| 11.35SaccadeControllers.H File Reference . . . . . | 2516 |
| 11.35Saccades.C File Reference . . . . .           | 2518 |
| 11.35SafeCopy.H File Reference . . . . .           | 2520 |
| 11.35Sal_count.C File Reference . . . . .          | 2522 |
| 11.35Saliency.C File Reference . . . . .           | 2524 |
| 11.35Saliency.H File Reference . . . . .           | 2527 |
| 11.35SaliencyMap.C File Reference . . . . .        | 2540 |

|        |                                                           |      |
|--------|-----------------------------------------------------------|------|
| 11.366 | <a href="#">SaliencyMap.H File Reference</a>              | 2541 |
| 11.365 | <a href="#">Sbqa.C File Reference</a>                     | 2543 |
| 11.362 | <a href="#">ScaleSpace.C File Reference</a>               | 2544 |
| 11.363 | <a href="#">ScaleSpace.H File Reference</a>               | 2546 |
| 11.364 | <a href="#">Searcharray.C File Reference</a>              | 2548 |
| 11.365 | <a href="#">Searcharray2.C File Reference</a>             | 2550 |
| 11.366 | <a href="#">Searcharray2d.C File Reference</a>            | 2552 |
| 11.367 | <a href="#">SegmentImage.C File Reference</a>             | 2554 |
| 11.368 | <a href="#">SegmentImage.H File Reference</a>             | 2555 |
| 11.369 | <a href="#">SegmentImageMerge.H File Reference</a>        | 2556 |
| 11.370 | <a href="#">SegmentImageTrackMC.H File Reference</a>      | 2558 |
| 11.371 | <a href="#">Serial.C File Reference</a>                   | 2559 |
| 11.372 | <a href="#">Serial.H File Reference</a>                   | 2560 |
| 11.373 | <a href="#">ShapeEstimator.C File Reference</a>           | 2562 |
| 11.374 | <a href="#">ShapeEstimator.H File Reference</a>           | 2564 |
| 11.375 | <a href="#">ShapeEstimatorModes.H File Reference</a>      | 2565 |
| 11.376 | <a href="#">ShapeEstimatorWebpage.C File Reference</a>    | 2568 |
| 11.377 | <a href="#">SharedPtr.H File Reference</a>                | 2570 |
| 11.378 | <a href="#">ShowFilters.C File Reference</a>              | 2574 |
| 11.379 | <a href="#">SimulationViewer.C File Reference</a>         | 2576 |
| 11.380 | <a href="#">SimulationViewer.H File Reference</a>         | 2577 |
| 11.381 | <a href="#">SimulationViewerCompress.C File Reference</a> | 2579 |
| 11.382 | <a href="#">SimulationViewerCompress.H File Reference</a> | 2582 |
| 11.383 | <a href="#">SimulationViewerEyeMvt.C File Reference</a>   | 2583 |
| 11.384 | <a href="#">SimulationViewerEyeMvt.H File Reference</a>   | 2585 |
| 11.385 | <a href="#">SimulationViewerEyeMvt2.C File Reference</a>  | 2586 |
| 11.386 | <a href="#">SimulationViewerEyeMvt2.H File Reference</a>  | 2588 |
| 11.387 | <a href="#">SimulationViewerStd.C File Reference</a>      | 2589 |
| 11.388 | <a href="#">SimulationViewerStd.H File Reference</a>      | 2591 |
| 11.389 | <a href="#">SingleChannel.C File Reference</a>            | 2593 |
| 11.390 | <a href="#">SingleChannel.H File Reference</a>            | 2595 |

|                                                        |      |
|--------------------------------------------------------|------|
| 11.39SingleChannelBeo.C File Reference . . . . .       | 2597 |
| 11.39SingleChannelBeo.H File Reference . . . . .       | 2598 |
| 11.39SingleChannelBeoServer.C File Reference . . . . . | 2600 |
| 11.39SingleChannelBeoServer.H File Reference . . . . . | 2601 |
| 11.39SingleChannelSurprise.C File Reference . . . . .  | 2602 |
| 11.39SingleChannelSurprise.H File Reference . . . . .  | 2603 |
| 11.39SockServ.C File Reference . . . . .               | 2604 |
| 11.39SockServ.H File Reference . . . . .               | 2606 |
| 11.39SoxChannel.C File Reference . . . . .             | 2608 |
| 11.40SoxChannel.H File Reference . . . . .             | 2609 |
| 11.40SpeedSensor.C File Reference . . . . .            | 2610 |
| 11.40SpeedSensor.H File Reference . . . . .            | 2611 |
| 11.40Sc.C File Reference . . . . .                     | 2612 |
| 11.40Sc.H File Reference . . . . .                     | 2613 |
| 11.40Staircase.C File Reference . . . . .              | 2615 |
| 11.40Staircase.H File Reference . . . . .              | 2616 |
| 11.40Stats.C File Reference . . . . .                  | 2617 |
| 11.40Stats.H File Reference . . . . .                  | 2618 |
| 11.40StatsIncl.H File Reference . . . . .              | 2619 |
| 11.41StdBrain.C File Reference . . . . .               | 2621 |
| 11.41StdBrain.H File Reference . . . . .               | 2623 |
| 11.41Tim-noise.C File Reference . . . . .              | 2626 |
| 11.41Tim-surprise.C File Reference . . . . .           | 2627 |
| 11.41Streamvision.C File Reference . . . . .           | 2628 |
| 11.41Streamvision2.C File Reference . . . . .          | 2629 |
| 11.41StringConversions.C File Reference . . . . .      | 2631 |
| 11.41StringConversions.H File Reference . . . . .      | 2636 |
| 11.41SurpriseImage.C File Reference . . . . .          | 2642 |
| 11.41SurpriseImage.H File Reference . . . . .          | 2643 |
| 11.42SurpriseMap.C File Reference . . . . .            | 2645 |
| 11.42SurpriseMap.H File Reference . . . . .            | 2646 |



|        |                                                  |      |
|--------|--------------------------------------------------|------|
| 11.423 | SurpriseModel.C File Reference . . . . .         | 2648 |
| 11.423 | SurpriseModel.H File Reference . . . . .         | 2651 |
| 11.424 | TaskRelevanceMap.C File Reference . . . . .      | 2653 |
| 11.425 | TaskRelevanceMap.H File Reference . . . . .      | 2655 |
| 11.426 | TCPLiServ.C File Reference . . . . .             | 2657 |
| 11.427 | TCPLiServ.H File Reference . . . . .             | 2659 |
| 11.428 | TCPcommunicator.C File Reference . . . . .       | 2661 |
| 11.429 | TCPcommunicator.H File Reference . . . . .       | 2663 |
| 11.430 | TCPdefs.H File Reference . . . . .               | 2665 |
| 11.431 | TCPmessage.C File Reference . . . . .            | 2667 |
| 11.431 | TCPmessage.H File Reference . . . . .            | 2669 |
| 11.432 | Test-align.C File Reference . . . . .            | 2671 |
| 11.434 | Test-Anim.C File Reference . . . . .             | 2672 |
| 11.435 | Test-audioGrab.C File Reference . . . . .        | 2675 |
| 11.436 | Test-Beobot.C File Reference . . . . .           | 2677 |
| 11.437 | Test-BeoChip.C File Reference . . . . .          | 2678 |
| 11.438 | Test-beoGrab-master.C File Reference . . . . .   | 2679 |
| 11.439 | Test-BeoSub.C File Reference . . . . .           | 2680 |
| 11.440 | Test-BeoSubMotor.C File Reference . . . . .      | 2681 |
| 11.441 | Test-binMorph.C File Reference . . . . .         | 2682 |
| 11.442 | Test-BitObject.C File Reference . . . . .        | 2683 |
| 11.443 | Test-BitObjectExtract.C File Reference . . . . . | 2684 |
| 11.444 | Test-BPnnet.C File Reference . . . . .           | 2685 |
| 11.445 | Test-cameraMotion.C File Reference . . . . .     | 2690 |
| 11.446 | Test-carcontrol.C File Reference . . . . .       | 2692 |
| 11.447 | Test-color.C File Reference . . . . .            | 2694 |
| 11.448 | Test-Colors.C File Reference . . . . .           | 2695 |
| 11.449 | Test-colorSegment.C File Reference . . . . .     | 2697 |
| 11.450 | Test-Compare.C File Reference . . . . .          | 2699 |
| 11.451 | Test-component.C File Reference . . . . .        | 2700 |
| 11.452 | Test-compress.C File Reference . . . . .         | 2701 |

|        |                                                   |      |
|--------|---------------------------------------------------|------|
| 11.453 | test-covEstimate.C File Reference . . . . .       | 2705 |
| 11.454 | test-diskgrab1394.C File Reference . . . . .      | 2707 |
| 11.455 | test-dyncast.C File Reference . . . . .           | 2710 |
| 11.456 | test-EdgeDetect.C File Reference . . . . .        | 2711 |
| 11.457 | test-FFN.C File Reference . . . . .               | 2712 |
| 11.458 | test-flood.C File Reference . . . . .             | 2715 |
| 11.459 | test-followColor-slave.C File Reference . . . . . | 2716 |
| 11.460 | test-followColor.C File Reference . . . . .       | 2718 |
| 11.461 | test-Foveator.C File Reference . . . . .          | 2721 |
| 11.462 | test-gabor.C File Reference . . . . .             | 2723 |
| 11.463 | test-gabor2.C File Reference . . . . .            | 2724 |
| 11.464 | test-gabor3.C File Reference . . . . .            | 2726 |
| 11.465 | test-gabor4.C File Reference . . . . .            | 2728 |
| 11.466 | test-gist.C File Reference . . . . .              | 2730 |
| 11.467 | test-GPS.C File Reference . . . . .               | 2732 |
| 11.468 | test-grab.C File Reference . . . . .              | 2733 |
| 11.469 | test-gyro.C File Reference . . . . .              | 2735 |
| 11.470 | test-hmax.C File Reference . . . . .              | 2736 |
| 11.471 | test-hmax2.C File Reference . . . . .             | 2739 |
| 11.472 | test-hmax3.C File Reference . . . . .             | 2741 |
| 11.473 | test-hmax4.C File Reference . . . . .             | 2743 |
| 11.474 | test-HSV.C File Reference . . . . .               | 2744 |
| 11.475 | test-hueChannel.C File Reference . . . . .        | 2746 |
| 11.476 | test-Image.C File Reference . . . . .             | 2747 |
| 11.477 | test-ImageDisplay.C File Reference . . . . .      | 2750 |
| 11.478 | test-ImageEqual.C File Reference . . . . .        | 2751 |
| 11.479 | test-Jet.C File Reference . . . . .               | 2752 |
| 11.480 | test-junction.C File Reference . . . . .          | 2754 |
| 11.481 | test-KalmanFilter.C File Reference . . . . .      | 2755 |
| 11.482 | test-KeyBoard.C File Reference . . . . .          | 2756 |
| 11.483 | test-keyboardGrab.C File Reference . . . . .      | 2757 |

|        |                                                 |      |
|--------|-------------------------------------------------|------|
| 11.484 | test-LevelSpec.C File Reference . . . . .       | 2759 |
| 11.485 | test-mbari.C File Reference . . . . .           | 2760 |
| 11.486 | test-monty.C File Reference . . . . .           | 2762 |
| 11.487 | test-MPEGStream.C File Reference . . . . .      | 2763 |
| 11.488 | test-multigrab.C File Reference . . . . .       | 2764 |
| 11.489 | test-multiTrack.C File Reference . . . . .      | 2766 |
| 11.490 | test-NPclassify.C File Reference . . . . .      | 2768 |
| 11.491 | test-NPclassify2.C File Reference . . . . .     | 2772 |
| 11.492 | test-optimizeHSV.C File Reference . . . . .     | 2774 |
| 11.493 | test-Pixels.C File Reference . . . . .          | 2776 |
| 11.494 | test-psycho.C File Reference . . . . .          | 2777 |
| 11.495 | test-radioDecoder.C File Reference . . . . .    | 2778 |
| 11.496 | test-raomodel.C File Reference . . . . .        | 2780 |
| 11.497 | test-Raster.C File Reference . . . . .          | 2782 |
| 11.498 | test-reset.C File Reference . . . . .           | 2784 |
| 11.499 | test-ShapeEstimator.C File Reference . . . . .  | 2785 |
| 11.500 | test-shift.C File Reference . . . . .           | 2788 |
| 11.501 | test-smcompress.C File Reference . . . . .      | 2789 |
| 11.502 | test-SoxChannel.C File Reference . . . . .      | 2794 |
| 11.503 | test-SpeedSensor.C File Reference . . . . .     | 2796 |
| 11.504 | test-ssc.C File Reference . . . . .             | 2797 |
| 11.505 | test-stereo.C File Reference . . . . .          | 2799 |
| 11.506 | test-surprise.C File Reference . . . . .        | 2800 |
| 11.507 | test-VCC4.C File Reference . . . . .            | 2801 |
| 11.508 | test-Xwin.C File Reference . . . . .            | 2803 |
| 11.509 | testimagesat.C File Reference . . . . .         | 2804 |
| 11.510 | TestSuite.C File Reference . . . . .            | 2805 |
| 11.511 | TestSuite.H File Reference . . . . .            | 2806 |
| 11.512 | Timer.H File Reference . . . . .                | 2808 |
| 11.513 | train-gist_headSize.C File Reference . . . . .  | 2811 |
| 11.514 | train-gist_headSize2.C File Reference . . . . . | 2816 |

|                                                           |      |
|-----------------------------------------------------------|------|
| 11.515TypeTraits.H File Reference . . . . .               | 2819 |
| 11.5164Lgrabber.C File Reference . . . . .                | 2820 |
| 11.5174Lgrabber.H File Reference . . . . .                | 2822 |
| 11.5184LgrabberLight.C File Reference . . . . .           | 2825 |
| 11.5194LgrabberLight.H File Reference . . . . .           | 2826 |
| 11.520CC4.C File Reference . . . . .                      | 2827 |
| 11.521VCC4.H File Reference . . . . .                     | 2828 |
| 11.522VCC4cmddef.H File Reference . . . . .               | 2830 |
| 11.523ideodiff.C File Reference . . . . .                 | 2834 |
| 11.524ideograb.C File Reference . . . . .                 | 2835 |
| 11.525ision.C File Reference . . . . .                    | 2837 |
| 11.526ision.H File Reference . . . . .                    | 2844 |
| 11.527VisualBuffer.C File Reference . . . . .             | 2846 |
| 11.528VisualBuffer.H File Reference . . . . .             | 2848 |
| 11.529VisualCortex.C File Reference . . . . .             | 2849 |
| 11.530VisualCortex.H File Reference . . . . .             | 2851 |
| 11.531VisualCortexBeo.C File Reference . . . . .          | 2854 |
| 11.532VisualCortexBeo.H File Reference . . . . .          | 2855 |
| 11.533VisualCortexConfigurator.C File Reference . . . . . | 2856 |
| 11.534VisualCortexConfigurator.H File Reference . . . . . | 2857 |
| 11.535VisualCortexEyeMvt.C File Reference . . . . .       | 2858 |
| 11.536VisualCortexEyeMvt.H File Reference . . . . .       | 2859 |
| 11.537VisualCortexSurprise.C File Reference . . . . .     | 2860 |
| 11.538VisualCortexSurprise.H File Reference . . . . .     | 2861 |
| 11.539VisualEvent.C File Reference . . . . .              | 2862 |
| 11.540VisualEvent.H File Reference . . . . .              | 2864 |
| 11.541VisualFeatures.H File Reference . . . . .           | 2866 |
| 11.542VisualObject.C File Reference . . . . .             | 2871 |
| 11.543VisualObject.H File Reference . . . . .             | 2872 |
| 11.544VisualObjectDB.C File Reference . . . . .           | 2874 |
| 11.545VisualObjectDB.H File Reference . . . . .           | 2875 |

|           |                                                            |             |
|-----------|------------------------------------------------------------|-------------|
| 11.546    | WebPageHelpers.C File Reference . . . . .                  | 2877        |
| 11.547    | WebPageHelpers.H File Reference . . . . .                  | 2881        |
| 11.548    | WinnerTakeAll.C File Reference . . . . .                   | 2885        |
| 11.549    | WinnerTakeAll.H File Reference . . . . .                   | 2886        |
| 11.550    | WTawinner.H File Reference . . . . .                       | 2888        |
| 11.551    | KWindow.C File Reference . . . . .                         | 2889        |
| 11.552    | XWindow.H File Reference . . . . .                         | 2890        |
| 11.553    | XWinManaged.C File Reference . . . . .                     | 2893        |
| 11.554    | XWinManaged.H File Reference . . . . .                     | 2894        |
| <b>12</b> | <b>iLab Neuromorphic Vision Toolkit Page Documentation</b> | <b>2897</b> |
| 12.1      | Building models using the toolkit . . . . .                | 2897        |
| 12.2      | Changes from Revisions 3.0 to 3.1 in src3/ . . . . .       | 2902        |
| 12.3      | Towards a neural simulation environment . . . . .          | 2903        |
| 12.4      | Tips and frequently asked questions . . . . .              | 2905        |
| 12.5      | Changes from src2 to src3 . . . . .                        | 2911        |
| 12.6      | The make/build process . . . . .                           | 2912        |
| 12.7      | Ref-counting and copy-on-write . . . . .                   | 2914        |
| 12.8      | Image iterators . . . . .                                  | 2919        |
| 12.9      | Free functions for image operations . . . . .              | 2922        |
| 12.10     | Test suite(s) . . . . .                                    | 2924        |
| 12.11     | The std:: namespace . . . . .                              | 2925        |
| 12.12     | Use of std::numeric_limits . . . . .                       | 2927        |



## Chapter 1

# iLab Neuromorphic Vision Toolkit Main Page

Welcome to the iLab Neuromorphic Vision C++ Toolkit. This toolkit is being developed at USC and Caltech to provide a comprehensive suite of C++ classes for image processing, biologically-inspired computational vision, hardware interfacing, and parallel processing on Linux beowulf clusters. Among the executables included in the distribution are our bottom-up, saliency-based visual attention model (single-CPU and beowulf versions), models of contour integration in primary visual cortex, models of object recognition in inferotemporal cortex, and our beobots quad-CPU autonomous neuromorphic robot code. Please see the [README](#) for installation and compilation instructions.

- **src/** is an obsolete source tree. It was active approximately between 1995 (not under CVS control) and late 2000, with minor compatibility updates throughout year 2001. When development in `src/` began, support for C++ templates in `g++` was incomplete, so `src/` relies on scripts to generate instantiations of source files from a generic source, simulating template behavior. The move from `src/` to `src2/` mainly includes switching the source tree to using templates.
- **src2/** is an obsolete source tree as well. During 2001, Rob Peters and others started introducing a number of powerful new components to the toolkit, which generated sufficient incompatibility to warrant a move to a new source tree, `src3/`. See [here](#) for notes on the changes to `src3` relative to `src2`.
- **src3/** is the current active source tree. In early 2003, some fairly large changes were introduced, but to avoid losing the whole CVS history once again, we kept the code in `src3/` rather than moving to `src4`. See [here](#) for notes on the changes in revision 3.1 relative to revision 3.0.

The main program to use is called **ezvision** in the saliency/bin/ directory. See the [README](#) for examples of how to use it.

See [here](#) for tips on how to build models using components of this toolkit.

To further help you navigate the source tree, see the available [CVSweb browser](#) (use the same username and password as you used to access this page).



# Chapter 2

## iLab Neuromorphic Vision Toolkit Module Index

### 2.1 iLab Neuromorphic Vision Toolkit Modules

Here is a list of all modules:

|                                        |    |
|----------------------------------------|----|
| max-normalization algorithms . . . . . | 51 |
| String conversion helpers . . . . .    | 61 |



# Chapter 3

## iLab Neuromorphic Vision Toolkit Namespace Index

### 3.1 iLab Neuromorphic Vision Toolkit Namespace List

Here is a list of all documented namespaces with brief descriptions:

|                                                                                            |                    |
|--------------------------------------------------------------------------------------------|--------------------|
| <a href="#">Png</a> (Namespace for functions related to PNG-format image files ) . . . . . | <a href="#">75</a> |
| <a href="#">Raster</a> (For reading, writing and displaying images ) . . . . .             | <a href="#">77</a> |
| <a href="#">std</a> (The Standard Template Library (STL) ) . . . . .                       | <a href="#">84</a> |



## Chapter 4

# iLab Neuromorphic Vision Toolkit Hierarchical Index

### 4.1 iLab Neuromorphic Vision Toolkit Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

|                                          |     |
|------------------------------------------|-----|
| _DisplayModel                            |     |
| _GlobalData                              |     |
| _IterationData                           |     |
| _MeasData                                |     |
| _ObservationModel                        |     |
| _PriorModel                              |     |
| _ProcessModel                            |     |
| _SceneModel                              |     |
| A                                        |     |
| Accum                                    |     |
| Animate                                  |     |
| AppContext . . . . .                     | 85  |
| Args                                     |     |
| ArrayData< T > . . . . .                 | 87  |
| ArrayData< float > . . . . .             | 87  |
| ArrayData< LeakyIntegrator > . . . . .   | 87  |
| ArrayData< LeakyIntFire > . . . . .      | 87  |
| ArrayHandle< T > . . . . .               | 93  |
| ArrayHandle< float > . . . . .           | 93  |
| ArrayHandle< LeakyIntegrator > . . . . . | 93  |
| ArrayHandle< LeakyIntFire > . . . . .    | 93  |
| ArrayHelper< T, isTrivial > . . . . .    | 97  |
| ArrayHelper< T, true > . . . . .         | 100 |

|                                         |     |
|-----------------------------------------|-----|
| atomic_int_t                            |     |
| B                                       |     |
| Beobot . . . . .                        | 125 |
| BeobotAction . . . . .                  | 130 |
| BeobotMemory . . . . .                  | 137 |
| BeobotSensation . . . . .               | 139 |
| BeobotSensors . . . . .                 | 143 |
| BeobotVisualCortex . . . . .            | 144 |
| BeoChipListener . . . . .               | 169 |
| MyBeoChipListener . . . . .             | 940 |
| BeoSubAction . . . . .                  | 181 |
| BeoSubActionDive . . . . .              | 187 |
| BeoSubActionTurn . . . . .              | 190 |
| BestEntry . . . . .                     | 208 |
| binary< N >                             |     |
| binary< 0 >                             |     |
| BitObject . . . . .                     | 211 |
| blobProp                                |     |
| BPneuron . . . . .                      | 237 |
| BPnnnet . . . . .                       | 240 |
| bug . . . . .                           | 263 |
| Cam                                     |     |
| cameraConf . . . . .                    | 264 |
| CameraCoordTransforms                   |     |
| CameraParams                            |     |
| CamManager                              |     |
| carConf . . . . .                       | 278 |
| cascadeHold . . . . .                   | 290 |
| cfBank                                  |     |
| ChannelListener . . . . .               | 310 |
| cheb_series_struct                      |     |
| CheckedIterator< T > . . . . .          | 311 |
| CINNIC . . . . .                        | 317 |
| CINNICstatsRun                          |     |
| CloseButtonListener . . . . .           | 321 |
| ColorSignature                          |     |
| CondensationFilter                      |     |
| ContourConnection . . . . .             | 367 |
| ContourLayer . . . . .                  | 368 |
| ContourLayerDynamic . . . . .           | 369 |
| ContourLayerStatic . . . . .            | 370 |
| ContourNeuronCreate< TC2 > . . . . .    | 371 |
| ContourNeuronProp< CH, X, Y > . . . . . | 376 |
| contourRun . . . . .                    | 382 |
| convert_helper< T, TT, false, false >   |     |

|                                      |      |
|--------------------------------------|------|
| convert_helper< T, TT, false, true > |      |
| convert_helper< T, TT, true, false > |      |
| convert_helper< T, TT, true, true >  |      |
| covEstimate< T >                     |      |
| covHolder< T >                       |      |
| CpuTimer . . . . .                   | 393  |
| std::deque< T, Alloc > . . . . .     | 396  |
| Dims . . . . .                       | 397  |
| DispFlags                            |      |
| ProbDist::Distribution< T >          |      |
| DOMPrintFormatTarget                 |      |
| Edge . . . . .                       | 437  |
| Entity . . . . .                     | 448  |
| Entity_TWM . . . . .                 | 453  |
| EpsInv< T >                          |      |
| ExtTable                             |      |
| fCV_STM                              |      |
| FeatureVector . . . . .              | 467  |
| FeedForwardNetwork . . . . .         | 470  |
| FFTWWrapper                          |      |
| findColorIndex                       |      |
| FOEestimator . . . . .               | 489  |
| Foveator . . . . .                   | 491  |
| BlurFoveator . . . . .               | 234  |
| LPTFoveator . . . . .                | 781  |
| PyrFoveator . . . . .                | 1079 |
| FrameRange . . . . .                 | 509  |
| gaborElement                         |      |
| GACHromosome . . . . .               | 543  |
| GAPopulation . . . . .               | 550  |
| ProbDist::Gaussian< T >              |      |
| GPSLocation                          |      |
| GPSPvtDataType                       |      |
| Gyro                                 |      |
| Heading . . . . .                    | 581  |
| Histogram . . . . .                  | 584  |
| Hmax . . . . .                       | 587  |
| HREntry . . . . .                    | 596  |
| HyperRectangle . . . . .             | 604  |
| Image< T > . . . . .                 | 611  |
| ImageSpring< T > . . . . .           | 684  |
| Jet< T > . . . . .                   | 716  |
| SurpriseImage< T > . . . . .         | 1417 |
| Image< float > . . . . .             | 611  |
| SaliencyMapFast . . . . .            | 1179 |

|                                      |      |
|--------------------------------------|------|
| SaliencyMapTrivial . . . . .         | 1193 |
| TaskRelevanceMapKillN . . . . .      | 1451 |
| TaskRelevanceMapKillStatic . . . . . | 1456 |
| TaskRelevanceMapStd . . . . .        | 1462 |
| WinnerTakeAllDummy . . . . .         | 1676 |
| WinnerTakeAllFast . . . . .          | 1681 |
| Image< LeakyIntegrator > . . . . .   | 611  |
| SaliencyMapStd . . . . .             | 1186 |
| Image< LeakyIntFire > . . . . .      | 611  |
| WinnerTakeAllStd . . . . .           | 1686 |
| ImageCache< T > . . . . .            | 657  |
| ImageCacheAvg< T > . . . . .         | 664  |
| ImageCacheMinMax< T > . . . . .      | 668  |
| ImageMap . . . . .                   | 676  |
| ImageSet< T > . . . . .              | 677  |
| InputLine                            |      |
| InstanceCounter                      |      |
| IRoom_AgentMinder                    |      |
| IRoom_CameraManager                  |      |
| IRoom_ExampleModule                  |      |
| IRoom_PursuitTracker                 |      |
| IRoom_SignatureRepository            |      |
| IRoom_UserInterface                  |      |
| IRoomAction                          |      |
| IRoomAgentInfo                       |      |
| IRoomCallJavaMsg                     |      |
| IRoomCameraInfo                      |      |
| IRoomCommandLine                     |      |
| IRoomFindSignalCamera                |      |
| IRoomHandoff                         |      |
| IRoomMoveSignal                      |      |
| IRoomResource                        |      |
| IRoomResourceReport                  |      |
| IRoomSaccade                         |      |
| JetSpec . . . . .                    | 723  |
| KalmanFilter . . . . .               | 736  |
| KDTree . . . . .                     | 741  |
| KeyBoard . . . . .                   | 744  |
| Keypoint                             |      |
| KeypointMatch . . . . .              | 746  |
| KnowledgeBase . . . . .              | 749  |
| LeakyIntegrator . . . . .            | 760  |
| LeakyIntFire . . . . .               | 763  |
| LevelSpec . . . . .                  | 766  |
| LGraph                               |      |



|                                                 |      |
|-------------------------------------------------|------|
| std::list< T, Alloc > . . . . .                 | 773  |
| locn                                            |      |
| Map . . . . .                                   | 787  |
| std::map< Key, Data, Compare, Alloc > . . . . . | 791  |
| Matrix3 . . . . .                               | 792  |
| Raster::MetaData . . . . .                      | 812  |
| MexBuf . . . . .                                | 817  |
| MexBufInit . . . . .                            | 818  |
| MexReturn . . . . .                             | 824  |
| ModelComponent . . . . .                        | 827  |
| Agent                                           |      |
| AttentionGuidanceMap . . . . .                  | 102  |
| AttentionGuidanceMapStd . . . . .               | 112  |
| AttentionGuidanceMapConfigurator . . . . .      | 108  |
| AudioGrabber . . . . .                          | 116  |
| AudioMixer . . . . .                            | 121  |
| BeobotEffectors . . . . .                       | 134  |
| BeoChip . . . . .                               | 151  |
| BeoSub . . . . .                                | 171  |
| BeoSubDepthSensor . . . . .                     | 193  |
| BeoSubMotor . . . . .                           | 196  |
| Beowulf . . . . .                               | 200  |
| Brain . . . . .                                 | 243  |
| GetSaliency . . . . .                           | 559  |
| StdBrain . . . . .                              | 1402 |
| CameraControl . . . . .                         | 273  |
| CarControl . . . . .                            | 283  |
| ChannelBase . . . . .                           | 293  |
| ComplexChannel . . . . .                        | 340  |
| ColorChannel . . . . .                          | 331  |
| ColorChannelBeo . . . . .                       | 335  |
| ColorChannelSurprise . . . . .                  | 337  |
| CompositeColorChannel . . . . .                 | 362  |
| CompositeColorChannelSurprise . . . . .         | 365  |
| EndPointChannel . . . . .                       | 441  |
| EndPointChannelSurprise . . . . .               | 445  |
| LJunctionChannel . . . . .                      | 774  |
| LJunctionChannelSurprise . . . . .              | 778  |
| MotionChannel . . . . .                         | 903  |
| MotionChannelBeo . . . . .                      | 907  |
| MotionChannelSurprise . . . . .                 | 909  |
| MultiColorBandChannel . . . . .                 | 919  |
| MultiColorBandChannelBeo . . . . .              | 923  |
| MultiColorBandChannelSurprise . . . . .         | 925  |

|                                        |      |
|----------------------------------------|------|
| MultiConvolveChannel . . . . .         | 928  |
| MultiConvolveChannelSurprise . . . . . | 932  |
| MultiDirectFeedChannel . . . . .       | 934  |
| OrientationChannel . . . . .           | 962  |
| OrientationChannelBeo . . . . .        | 969  |
| OrientationChannelSurprise . . . . .   | 971  |
| SOCColorChannel . . . . .              | 1352 |
| SOCColorChannelSurprise . . . . .      | 1359 |
| SoxChannel . . . . .                   | 1378 |
| StereoChannel . . . . .                | 1405 |
| TJunctionChannel . . . . .             | 1523 |
| TJunctionChannelSurprise . . . . .     | 1527 |
| XJunctionChannel . . . . .             | 1696 |
| XJunctionChannelSurprise . . . . .     | 1700 |
| ContourChannel                         |      |
| DirectFeedChannel . . . . .            | 402  |
| SingleChannel . . . . .                | 1297 |
| BlueChannel . . . . .                  | 222  |
| BlueChannelSurprise . . . . .          | 224  |
| BlueYellowChannel . . . . .            | 227  |
| BlueYellowChannelBeo . . . . .         | 229  |
| BlueYellowChannelSurprise . . . . .    | 231  |
| ColorBandChannel . . . . .             | 324  |
| ColorBandChannelBeo . . . . .          | 326  |
| ColorBandChannelSurprise . . . . .     | 328  |
| ConvolveChannel . . . . .              | 386  |
| ConvolveChannelSurprise . . . . .      | 388  |
| DirectionChannel . . . . .             | 415  |
| DirectionChannelBeo . . . . .          | 418  |
| DirectionChannelSurprise . . . . .     | 420  |
| DisparityChannel . . . . .             | 425  |
| DummyChannel . . . . .                 | 430  |
| DummyChannelSurprise . . . . .         | 434  |
| EntropyChannel . . . . .               | 457  |
| FlickerChannel . . . . .               | 479  |
| FlickerChannelBeo . . . . .            | 483  |
| FlickerChannelSurprise . . . . .       | 486  |
| GaborChannel . . . . .                 | 529  |
| GaborChannelBeo . . . . .              | 533  |
| GaborChannelSurprise . . . . .         | 536  |
| GreenChannel . . . . .                 | 576  |
| GreenChannelSurprise . . . . .         | 578  |
| HueChannel . . . . .                   | 599  |
| SkinHueChannel . . . . .               | 1337 |

|                                        |      |
|----------------------------------------|------|
| InformationChannel . . . . .           | 699  |
| IntensityChannel . . . . .             | 708  |
| IntensityChannelBeo . . . . .          | 710  |
| IntensityChannelSurprise . . . . .     | 713  |
| JunctionChannel . . . . .              | 729  |
| JunctionChannelSurprise . . . . .      | 733  |
| PN03contrastChannel . . . . .          | 1018 |
| RedChannel . . . . .                   | 1108 |
| RedChannelSurprise . . . . .           | 1110 |
| RedGreenChannel . . . . .              | 1113 |
| RedGreenChannelBeo . . . . .           | 1115 |
| RedGreenChannelSurprise . . . . .      | 1118 |
| RGBConvolveChannel . . . . .           | 1141 |
| ScorrChannel . . . . .                 | 1201 |
| SOBlueYellowChannel . . . . .          | 1340 |
| SOBlueYellowChannelSurprise . . . . .  | 1342 |
| SOGreenRedChannel . . . . .            | 1364 |
| SOGreenRedChannelSurprise . . . . .    | 1366 |
| SORedGreenChannel . . . . .            | 1369 |
| SORedGreenChannelSurprise . . . . .    | 1371 |
| SOYellowBlueChannel . . . . .          | 1384 |
| SOYellowBlueChannelSurprise . . . . .  | 1386 |
| TcorrChannel . . . . .                 | 1468 |
| TemplateMatchChannel . . . . .         | 1494 |
| PedestrianChannel . . . . .            | 996  |
| PedestrianChannelSurprise . . . . .    | 999  |
| TemplateMatchChannelSurprise . . . . . | 1497 |
| VarianceChannel . . . . .              | 1556 |
| YellowChannel . . . . .                | 1715 |
| YellowChannelSurprise . . . . .        | 1717 |
| COMPort . . . . .                      | 356  |
| DirTest . . . . .                      | 423  |
| featureClusterVision                   |      |
| FrameGrabber . . . . .                 | 496  |
| IEEE1394grabber . . . . .              | 607  |
| V4Lgrabber . . . . .                   | 1548 |
| FrameGrabberConfigurator . . . . .     | 505  |
| FrameSeries . . . . .                  | 512  |
| InputFrameSeries . . . . .             | 703  |
| OutputFrameSeries . . . . .            | 976  |
| GistEstimator . . . . .                | 563  |
| GistEstimatorConfigurator . . . . .    | 566  |
| GpsDev . . . . .                       | 570  |
| HMR3300 . . . . .                      | 593  |

|                                              |      |
|----------------------------------------------|------|
| ImageDisplay . . . . .                       | 671  |
| InferoTemporal . . . . .                     | 691  |
| InferoTemporalConfigurator . . . . .         | 695  |
| IRoom                                        |      |
| lcd . . . . .                                | 755  |
| MbariResultViewer . . . . .                  | 796  |
| ModelManager . . . . .                       | 854  |
| MexModelManager . . . . .                    | 819  |
| MotTest . . . . .                            | 912  |
| MPEGStream . . . . .                         | 914  |
| PsychoDisplay . . . . .                      | 1054 |
| RadioDecoder . . . . .                       | 1083 |
| Retina . . . . .                             | 1124 |
| RetinaStd . . . . .                          | 1134 |
| RetinaConfigurator . . . . .                 | 1130 |
| SaccadeController . . . . .                  | 1147 |
| EyeTrackerSaccadeController . . . . .        | 461  |
| FixedSaccadeController . . . . .             | 474  |
| FrictionSaccadeController . . . . .          | 523  |
| ModelEyeHeadController . . . . .             | 848  |
| ModelWrapperSaccadeController . . . . .      | 881  |
| Monkey2SaccadeController . . . . .           | 886  |
| MonkeyEyeHeadController . . . . .            | 891  |
| MonkeySaccadeController . . . . .            | 897  |
| SimpleEyeHeadSaccadeController . . . . .     | 1231 |
| ThresholdFrictionSaccadeController . . . . . | 1508 |
| ThresholdSaccadeController . . . . .         | 1513 |
| TrivialSaccadeController . . . . .           | 1537 |
| VisBufSaccadeController . . . . .            | 1588 |
| SaccadeControllerConfigurator . . . . .      | 1162 |
| SaliencyMap . . . . .                        | 1168 |
| SaliencyMapFast . . . . .                    | 1179 |
| SaliencyMapStd . . . . .                     | 1186 |
| SaliencyMapTrivial . . . . .                 | 1193 |
| SaliencyMapConfigurator . . . . .            | 1176 |
| Serial . . . . .                             | 1205 |
| ShapeEstimator . . . . .                     | 1216 |
| SimulationViewer . . . . .                   | 1235 |
| SimulationViewerCompress . . . . .           | 1242 |
| SimulationViewerEyeMvt . . . . .             | 1262 |
| SimulationViewerEyeMvt2 . . . . .            | 1272 |
| SimulationViewerStd . . . . .                | 1283 |
| SimulationViewerConfigurator . . . . .       | 1258 |
| SingleChannelBeoServer . . . . .             | 1327 |
| SingleChannelSurprise . . . . .              | 1331 |

|                                                      |      |
|------------------------------------------------------|------|
| SockServ . . . . .                                   | 1345 |
| SSC . . . . .                                        | 1390 |
| Staircase . . . . .                                  | 1396 |
| TaskRelevanceMap . . . . .                           | 1441 |
| TaskRelevanceMapKillIN . . . . .                     | 1451 |
| TaskRelevanceMapKillStatic . . . . .                 | 1456 |
| TaskRelevanceMapStd . . . . .                        | 1462 |
| TaskRelevanceMapConfigurator . . . . .               | 1447 |
| TCPcommunicator . . . . .                            | 1476 |
| VCC4 . . . . .                                       | 1559 |
| VisualBuffer . . . . .                               | 1592 |
| VisualCortex . . . . .                               | 1601 |
| VisualCortexBeo . . . . .                            | 1618 |
| VisualCortexEyeMvt . . . . .                         | 1626 |
| VisualCortexSurprise . . . . .                       | 1633 |
| VisualCortexConfigurator . . . . .                   | 1622 |
| WinnerTakeAll . . . . .                              | 1665 |
| WinnerTakeAllDummy . . . . .                         | 1676 |
| WinnerTakeAllFast . . . . .                          | 1681 |
| WinnerTakeAllStd . . . . .                           | 1686 |
| WinnerTakeAllConfigurator . . . . .                  | 1672 |
| ModelManager::ModelOption                            |      |
| ModelOptionDef . . . . .                             | 868  |
| ModelParamBase . . . . .                             | 875  |
| ModelParam< T > . . . . .                            | 871  |
| std::multimap< Key, Data, Compare, Alloc > . . . . . | 938  |
| std::multiset< Key, Compare, Alloc > . . . . .       | 939  |
| NewClass . . . . .                                   | 942  |
| NPclassify                                           |      |
| NPclassify2                                          |      |
| Object                                               |      |
| ObjectBase . . . . .                                 | 944  |
| OldFrameSeries . . . . .                             | 947  |
| Omni< T > . . . . .                                  | 948  |
| Ontology . . . . .                                   | 950  |
| OptionParser . . . . .                               | 953  |
| OptionSet . . . . .                                  | 958  |
| OptionSet::Impl                                      |      |
| Param                                                |      |
| ParamMap . . . . .                                   | 980  |
| ParamMap::Error . . . . .                            | 991  |
| ParamMap::Impl                                       |      |
| ParamMap::key_iterator                               |      |
| ParamMap::key_iterator::IterRep                      |      |
| PbmParser . . . . .                                  | 992  |

|                                          |      |
|------------------------------------------|------|
| PbmParser::Rep                           |      |
| PfmParser . . . . .                      | 1002 |
| PfmParser::Rep                           |      |
| PixelPoint                               |      |
| PixRGB< T > . . . . .                    | 1005 |
| PngParser . . . . .                      | 1021 |
| PngParser::Rep                           |      |
| PngWriter . . . . .                      | 1024 |
| Point< T, N > . . . . .                  | 1025 |
| Point2D . . . . .                        | 1029 |
| Point2DT . . . . .                       | 1039 |
| Point3D                                  |      |
| promote_trait< T1, T2 > . . . . .        | 1042 |
| promote_trait< byte, byte >              |      |
| promote_trait< byte, double >            |      |
| promote_trait< byte, float >             |      |
| promote_trait< byte, int16 >             |      |
| promote_trait< byte, int32 >             |      |
| promote_trait< float, double >           |      |
| promote_trait< int16, byte >             |      |
| promote_trait< int16, double >           |      |
| promote_trait< int16, float >            |      |
| promote_trait< int16, int16 >            |      |
| promote_trait< int16, int32 >            |      |
| promote_trait< int32, double >           |      |
| promote_trait< int32, float >            |      |
| promote_trait< PixRGB< byte >, byte >    |      |
| promote_trait< PixRGB< byte >, double >  |      |
| promote_trait< PixRGB< byte >, float >   |      |
| promote_trait< PixRGB< byte >, int16 >   |      |
| promote_trait< PixRGB< byte >, int32 >   |      |
| promote_trait< PixRGB< float >, double > |      |
| promote_trait< PixRGB< int16 >, byte >   |      |
| promote_trait< PixRGB< int16 >, double > |      |
| promote_trait< PixRGB< int16 >, float >  |      |
| promote_trait< PixRGB< int16 >, int16 >  |      |
| promote_trait< PixRGB< int16 >, int32 >  |      |
| promote_trait< PixRGB< int32 >, double > |      |
| promote_trait< PixRGB< int32 >, float >  |      |
| promoteFromTo< T1, T2 >                  |      |
| promoteFromTo< double, byte >            |      |
| promoteFromTo< double, float >           |      |
| promoteFromTo< double, int16 >           |      |
| promoteFromTo< double, int32 >           |      |
| promoteFromTo< float, byte >             |      |
| promoteFromTo< float, int16 >            |      |

|                                           |      |
|-------------------------------------------|------|
| promoteFromTo< float, int32 >             |      |
| promoteFromTo< int16, byte >              |      |
| promoteFromTo< int32, byte >              |      |
| promoteFromTo< int32, int16 >             |      |
| promoteFromTo< T, T >                     |      |
| Property . . . . .                        | 1043 |
| PropertyVectorSet . . . . .               | 1046 |
| PropHold . . . . .                        | 1049 |
| PsychoTask . . . . .                      | 1068 |
| PvmMessage . . . . .                      | 1070 |
| PyrBuilder< T > . . . . .                 | 1076 |
| ConvolvePyrBuilder< T > . . . . .         | 391  |
| GaborPyrBuilder< T > . . . . .            | 539  |
| GaborPyrBuilder< T > . . . . .            | 539  |
| GaussianPyrBuilder< T > . . . . .         | 555  |
| GenericPyrBuilder< T > . . . . .          | 557  |
| LaplacianPyrBuilder< T > . . . . .        | 753  |
| OrientedPyrBuilder< T > . . . . .         | 974  |
| ReichardtPyrBuilder< T > . . . . .        | 1121 |
| RGBConvolvePyrBuilder< T > . . . . .      | 1144 |
| PyrBuilder< float > . . . . .             | 1076 |
| TemplateMatchPyrBuilder . . . . .         | 1500 |
| Que_element                               |      |
| Randomizer< T >                           |      |
| Randomizer< PixRGB< T > >                 |      |
| Range< T > . . . . .                      | 1088 |
| readConfig . . . . .                      | 1093 |
| readMatrix . . . . .                      | 1101 |
| Rectangle . . . . .                       | 1102 |
| SafeCopyHelper< T, isTrivial > . . . . .  | 1166 |
| SafeCopyHelper< T, true > . . . . .       | 1167 |
| ScaleSpace . . . . .                      | 1198 |
| segmentImage                              |      |
| segmentImage2                             |      |
| segmentImageMC< T >                       |      |
| segmentImageMerge                         |      |
| segmentImageMerge2                        |      |
| segmentImageTrack                         |      |
| segmentImageTrack2                        |      |
| segmentImageTrackMC< T >                  |      |
| Serial2                                   |      |
| std::set< Key, Compare, Alloc > . . . . . | 1215 |
| SharedPtr< T > . . . . .                  | 1225 |
| Signature                                 |      |
| SignatureRepository                       |      |

|                                    |      |
|------------------------------------|------|
| SingleChannel::TPyr                |      |
| SingleChannelBeo . . . . .         | 1323 |
| BlueYellowChannelBeo . . . . .     | 229  |
| ColorBandChannelBeo . . . . .      | 326  |
| DirectionChannelBeo . . . . .      | 418  |
| FlickerChannelBeo . . . . .        | 483  |
| GaborChannelBeo . . . . .          | 533  |
| IntensityChannelBeo . . . . .      | 710  |
| RedGreenChannelBeo . . . . .       | 1115 |
| std::slist< T, Alloc > . . . . .   | 1339 |
| SomeInfo                           |      |
| SortedLimitedList< T > . . . . .   | 1374 |
| SoxWindow                          |      |
| SpeedSensor . . . . .              | 1389 |
| stats< T >                         |      |
| StraightLine2D . . . . .           | 1411 |
| std::string . . . . .              | 1416 |
| SurpriseMap< T > . . . . .         | 1422 |
| SurpriseModel . . . . .            | 1425 |
| SurpriseModelSG . . . . .          | 1431 |
| SurpriseModelSP . . . . .          | 1436 |
| TCPcliServ . . . . .               | 1472 |
| TCPmessage . . . . .               | 1485 |
| TempFile                           |      |
| TestData                           |      |
| TestSuite . . . . .                | 1502 |
| TestSuite::Impl                    |      |
| Timer . . . . .                    | 1519 |
| Token . . . . .                    | 1530 |
| ToolkitInit . . . . .              | 1535 |
| ToPow< T >                         |      |
| trackerData                        |      |
| Tree                               |      |
| Tree::Node                         |      |
| TWM . . . . .                      | 1540 |
| type_if< T, B > . . . . .          | 1544 |
| type_if< T, true >                 |      |
| type_with_N_bits< T, N > . . . . . | 1545 |
| TypeTraits< T > . . . . .          | 1546 |
| TypeTraits< bool >                 |      |
| TypeTraits< char >                 |      |
| TypeTraits< double >               |      |
| TypeTraits< float >                |      |
| TypeTraits< int >                  |      |
| TypeTraits< long >                 |      |



|                                     |      |
|-------------------------------------|------|
| TypeTraits< long double >           |      |
| TypeTraits< PixRGB< T > > . . . . . | 1547 |
| TypeTraits< short >                 |      |
| TypeTraits< unsigned char >         |      |
| TypeTraits< unsigned int >          |      |
| TypeTraits< unsigned long >         |      |
| TypeTraits< unsigned short >        |      |
| ProbDist::Uniform< T >              |      |
| ProbDist::UserDefined< T >          |      |
| V4LgrabberLight . . . . .           | 1552 |
| V4LPalette . . . . .                | 1554 |
| VCC4CMD                             |      |
| std::vector< T, Alloc > . . . . .   | 1571 |
| Vector2D . . . . .                  | 1572 |
| Vector3 . . . . .                   | 1581 |
| Vertex                              |      |
| VisualEvent . . . . .               | 1636 |
| VisualEventSet . . . . .            | 1643 |
| VisualObject . . . . .              | 1650 |
| VisualObjectDB . . . . .            | 1657 |
| VisualObjectMatch . . . . .         | 1661 |
| WayPoint                            |      |
| WTAwinner . . . . .                 | 1691 |
| XWindow . . . . .                   | 1703 |
| XWinManaged . . . . .               | 1708 |



## Chapter 5

# iLab Neuromorphic Vision Toolkit Class Index

### 5.1 iLab Neuromorphic Vision Toolkit Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

|                                                                                                                                              |     |
|----------------------------------------------------------------------------------------------------------------------------------------------|-----|
| <a href="#">AppContext</a> (Manages state information for vision.C::main() ) . . . . .                                                       | 85  |
| <a href="#">ArrayData&lt; T &gt;</a> (Implements a ref-counted 2-D array, should only be used<br>via <a href="#">ArrayHandle</a> ) . . . . . | 87  |
| <a href="#">ArrayHandle&lt; T &gt;</a> (This class provides ref-counting and copy-on-write for<br>ArrayData's ) . . . . .                    | 93  |
| <a href="#">ArrayHelper&lt; T, isTrivial &gt;</a> (This is a helper template struct for ArrayData's<br>constructors ) . . . . .              | 97  |
| <a href="#">ArrayHelper&lt; T, true &gt;</a> (Specialization of <a href="#">ArrayHelper</a> for builtin/trivial<br>types ) . . . . .         | 100 |
| <a href="#">AttentionGuidanceMap</a> (The task-relevance map base class ) . . . . .                                                          | 102 |
| <a href="#">AttentionGuidanceMapConfigurator</a> ( <a href="#">AttentionGuidanceMap</a> configurator ) .                                     | 108 |
| <a href="#">AttentionGuidanceMapStd</a> (The standard task-relevance map ) . . . . .                                                         | 112 |
| <a href="#">AudioGrabber</a> (Class to grab audio samples from an audio device ) . . . . .                                                   | 116 |
| <a href="#">AudioMixer</a> (Simple interface to the Linux audio mixer ) . . . . .                                                            | 121 |
| <a href="#">Beobot</a> (This class is a complete Beobot ) . . . . .                                                                          | 125 |
| <a href="#">BeobotAction</a> (This class defined motor actions for the Beobots ) . . . . .                                                   | 130 |
| <a href="#">BeobotEffectors</a> (This class contains all the effectors of a <a href="#">Beobot</a> ) . . . . .                               | 134 |
| <a href="#">BeobotMemory</a> (Memory of a <a href="#">Beobot</a> ) . . . . .                                                                 | 137 |
| <a href="#">BeobotSensation</a> (This class contains all the sensory informations but not the<br>action ) . . . . .                          | 139 |
| <a href="#">BeobotSensors</a> (All the sensors of a <a href="#">Beobot</a> ) . . . . .                                                       | 143 |
| <a href="#">BeobotVisualCortex</a> (Visual Cortex of a <a href="#">Beobot</a> ) . . . . .                                                    | 144 |
| <a href="#">BeoChip</a> ( <a href="#">BeoChip.H</a> Interface to Brian Hudson's BeoChip device ) . . . . .                                   | 151 |

|                                                                                                                                                    |     |
|----------------------------------------------------------------------------------------------------------------------------------------------------|-----|
| <a href="#">BeoChipListener</a> ( <a href="#">BeoChip</a> event listener ) . . . . .                                                               | 169 |
| <a href="#">BeoSub</a> (Definition and access functions for the <a href="#">BeoSub</a> ) . . . . .                                                 | 171 |
| <a href="#">BeoSubAction</a> (Helper base class for <a href="#">BeoSub</a> motor actions ) . . . . .                                               | 181 |
| <a href="#">BeoSubActionDive</a> (Helper base class for <a href="#">BeoSub</a> dive actions ) . . . . .                                            | 187 |
| <a href="#">BeoSubActionTurn</a> (Helper base class for <a href="#">BeoSub</a> turn actions ) . . . . .                                            | 190 |
| <a href="#">BeoSubDepthSensor</a> (Class for interfacing with a depth sensor ) . . . . .                                                           | 193 |
| <a href="#">BeoSubMotor</a> (Low-level driver for the <a href="#">BeoSub</a> motors ) . . . . .                                                    | 196 |
| <a href="#">Beowulf</a> (Simple interfacing to a Beowulf cluster ) . . . . .                                                                       | 200 |
| <a href="#">BestEntry</a> (The nearest entry in the tree to the target entry ) . . . . .                                                           | 208 |
| <a href="#">BitObject</a> (Object defined by a connected binary pixel region ) . . . . .                                                           | 211 |
| <a href="#">BlueChannel</a> (A blue color channel ) . . . . .                                                                                      | 222 |
| <a href="#">BlueChannelSurprise</a> (A blue color channel ) . . . . .                                                                              | 224 |
| <a href="#">BlueYellowChannel</a> (A blue-yellow double-opponent color channel ) . . . . .                                                         | 227 |
| <a href="#">BlueYellowChannelBeo</a> ( <a href="#">Beowulf</a> version of <a href="#">BlueYellowChannel</a> ) . . . . .                            | 229 |
| <a href="#">BlueYellowChannelSurprise</a> (A blue-yellow double-opponent color channel ) . . . . .                                                 | 231 |
| <a href="#">BlurFoveator</a> (A class that does space-variant processing with progressive blurring ) . . . . .                                     | 234 |
| <a href="#">BPneuron</a> (Neuron for a back prop neural net ( <a href="#">BPnnnet</a> ) ) . . . . .                                                | 237 |
| <a href="#">BPnnnet</a> (Describes structure of a 3 layer back prop neural net ) . . . . .                                                         | 240 |
| <a href="#">Brain</a> (This is the brain, a holder for various vision objects ) . . . . .                                                          | 243 |
| <a href="#">bug</a> (This gets thrown by the FATAL logs ) . . . . .                                                                                | 263 |
| <a href="#">cameraConf</a> (This class uses <a href="#">readConfig</a> to extract config parameters for <a href="#">CameraControl</a> ) . . . . .  | 264 |
| <a href="#">CameraControl</a> (This class is an interface to pan, tilt, and otherwise move the camera ) . . . . .                                  | 273 |
| <a href="#">carConf</a> (This class uses <a href="#">readConfig</a> to extract config parameters for <a href="#">CarControl</a> ) . . . . .        | 278 |
| <a href="#">CarControl</a> (This class is an interface to pilot the RC car ) . . . . .                                                             | 283 |
| <a href="#">cascadeHold</a> (This is a container class for holding cascading energy values ) . . . . .                                             | 290 |
| <a href="#">ChannelBase</a> ( <a href="#">ChannelBase</a> represents the various computational modules in <a href="#">VisualCortex</a> ) . . . . . | 293 |
| <a href="#">ChannelListener</a> (A callback class for use with <a href="#">ChannelBase</a> and its derivatives ) . . . . .                         | 310 |
| <a href="#">CheckedIterator&lt; T &gt;</a> (A range-checked iterator class for memory debugging ) . . . . .                                        | 311 |
| <a href="#">CINNIC</a> (Main class to run an image in CINNICtest for contours ) . . . . .                                                          | 317 |
| <a href="#">CloseButtonListener</a> (Watch for close button actions with all registered <a href="#">XWinManaged</a> windows ) . . . . .            | 321 |
| <a href="#">ColorBandChannel</a> (A color band channel ) . . . . .                                                                                 | 324 |
| <a href="#">ColorBandChannelBeo</a> ( <a href="#">Beowulf</a> version of <a href="#">ColorBandChannel</a> ) . . . . .                              | 326 |
| <a href="#">ColorBandChannelSurprise</a> (A color band channel ) . . . . .                                                                         | 328 |
| <a href="#">ColorChannel</a> (A double opponent color channel class that combines r/g, b/y subchannels ) . . . . .                                 | 331 |
| <a href="#">ColorChannelBeo</a> ( <a href="#">Beowulf</a> version of <a href="#">ColorChannel</a> ) . . . . .                                      | 335 |
| <a href="#">ColorChannelSurprise</a> (A double opponent color channel class that combines r/g, b/y subchannels ) . . . . .                         | 337 |

|                                                                                                                                |     |
|--------------------------------------------------------------------------------------------------------------------------------|-----|
| <a href="#">ComplexChannel</a> (A an abstract channel class that pools across several sub-channels ) . . . . .                 | 340 |
| <a href="#">COMPort</a> (A simple class for interfacing with a serial (COM) port ) . . . . .                                   | 356 |
| <a href="#">CompositeColorChannel</a> (A composite color channel class that contains single and double ! ) . . . . .           | 362 |
| <a href="#">CompositeColorChannelSurprise</a> (A composite color channel class that contains single and double opp ) . . . . . | 365 |
| <a href="#">ContourConnection</a> (The connection kernel between orientation-tuned units )                                     | 367 |
| <a href="#">ContourLayer</a> (Represents a single scale band in which contour facilitation is run ) . . . . .                  | 368 |
| <a href="#">ContourLayerDynamic</a> (Based on Achim Braun's unpublished Neural Computation paper ) . . . . .                   | 369 |
| <a href="#">ContourLayerStatic</a> (Based on <a href="#">CINNIC</a> (Nathan Mundhenk) ) . . . . .                              | 370 |
| <a href="#">ContourNeuronCreate&lt; TC2 &gt;</a> (This class will create the neuron connection template ) . . . . .            | 371 |
| <a href="#">ContourNeuronProp&lt; CH, X, Y &gt;</a> (Holds the property for each neuron ) . .                                  | 376 |
| <a href="#">contourRun</a> (Run the hyper column on a given image with a given connection template ) . . . . .                 | 382 |
| <a href="#">ConvolveChannel</a> (A generic convolution channel ) . . . . .                                                     | 386 |
| <a href="#">ConvolveChannelSurprise</a> (A generic convolution channel ) . . . . .                                             | 388 |
| <a href="#">ConvolvePyrBuilder&lt; T &gt;</a> (Builds pyramids based on arbitrary filters ) . .                                | 391 |
| <a href="#">CpuTimer</a> (Utility class for monitoring user/system cpu usage plus wall-clock time ) . . . . .                  | 393 |
| <a href="#">std::deque&lt; T, Alloc &gt;</a> (The STL deque container class. Use with <code>#include</code> ) . . . . .        | 396 |
| <a href="#">Dims</a> (A simple struct to hold a pair of width/height dimensions ) . . . . .                                    | 397 |
| <a href="#">DirectFeedChannel</a> (Direct Feed Channel ) . . . . .                                                             | 402 |
| <a href="#">DirectionChannel</a> (Motion sensitive channel with direction selectivity ) . . .                                  | 415 |
| <a href="#">DirectionChannelBeo</a> (Beowulf version of <a href="#">DirectionChannel</a> ) . . . . .                           | 418 |
| <a href="#">DirectionChannelSurprise</a> (Motion sensitive channel with direction selectivity ) . . . . .                      | 420 |
| <a href="#">DirTest</a> (Class for testing <a href="#">DirectionChannel</a> ) . . . . .                                        | 423 |
| <a href="#">DisparityChannel</a> (An disparity channel ) . . . . .                                                             | 425 |
| <a href="#">DummyChannel</a> (A dummy channel that simply passes its greyscale input to its output ) . . . . .                 | 430 |
| <a href="#">DummyChannelSurprise</a> (A dummy channel ) . . . . .                                                              | 434 |
| <a href="#">Edge</a> (Definition of Edge class and its functions ) . . . . .                                                   | 437 |
| <a href="#">EndPointChannel</a> (An end point detector channel that detects line ends at different ) . . . . .                 | 441 |
| <a href="#">EndPointChannelSurprise</a> (An end point detector channel that detects line ends at different ) . . . . .         | 445 |
| <a href="#">Entity</a> (Definition of Entity class and its functions ) . . . . .                                               | 448 |
| <a href="#">Entity_TWM</a> (Definition of Entity_TWM class and its functions ) . . . . .                                       | 453 |
| <a href="#">EntropyChannel</a> (An entropy channel ) . . . . .                                                                 | 457 |
| <a href="#">EyeTrackerSaccadeController</a> (EyeTracker saccade controller ) . . . . .                                         | 461 |
| <a href="#">FeatureVector</a> (Feature Vector ) . . . . .                                                                      | 467 |

|                                                                                                                             |     |
|-----------------------------------------------------------------------------------------------------------------------------|-----|
| <a href="#">FeedForwardNetwork</a> (A Feed Forward Network ) . . . . .                                                      | 470 |
| <a href="#">FixedSaccadeController</a> (Fixed saccade controller ) . . . . .                                                | 474 |
| <a href="#">FlickerChannel</a> (A temporal flicker channel ) . . . . .                                                      | 479 |
| <a href="#">FlickerChannelBeo</a> ( <a href="#">Beowulf</a> version of <a href="#">FlickerChannel</a> ) . . . . .           | 483 |
| <a href="#">FlickerChannelSurprise</a> (A temporal flicker channel ) . . . . .                                              | 486 |
| <a href="#">FOEestimator</a> (Compute the focus of expansion (FOE) from the pixel-based<br>optical flow ) . . . . .         | 489 |
| <a href="#">Foveator</a> (An abstract class for space-variant image processing ) . . . . .                                  | 491 |
| <a href="#">FrameGrabber</a> (Base class for a frame grabber device ) . . . . .                                             | 496 |
| <a href="#">FrameGrabberConfigurator</a> (This class is for run-time selection of a frame<br>grabber driver ) . . . . .     | 505 |
| <a href="#">FrameRange</a> (Specification of a range of frames with given inter-frame delay<br>(in ms) ) . . . . .          | 509 |
| <a href="#">FrameSeries</a> (FrameSeries manages a series of input or output frames ) . . . .                               | 512 |
| <a href="#">FrictionSaccadeController</a> (Friction-damped mass-spring saccade controller )                                 | 523 |
| <a href="#">GaborChannel</a> (An orientation-tuned gabor channel ) . . . . .                                                | 529 |
| <a href="#">GaborChannelBeo</a> ( <a href="#">Beowulf</a> version of <a href="#">GaborChannel</a> ) . . . . .               | 533 |
| <a href="#">GaborChannelSurprise</a> (An orientation-tuned gabor channel ) . . . . .                                        | 536 |
| <a href="#">GaborPyrBuilder&lt; T &gt;</a> (A pyramid based on convolution with oriented Ga-<br>bor filters ) . . . . .     | 539 |
| <a href="#">GACHromosome</a> (Chromosome class for genetic algorithm ) . . . . .                                            | 543 |
| <a href="#">GAPopulation</a> (Population class for genetic algorithm ) . . . . .                                            | 550 |
| <a href="#">GaussianPyrBuilder&lt; T &gt;</a> (Builds pyramids based on Gaussian filters ) . . .                            | 555 |
| <a href="#">GenericPyrBuilder&lt; T &gt;</a> (Builds pyramids of a type specified by a Pyramid-<br>Type ) . . . . .         | 557 |
| <a href="#">GetSaliency</a> (This class allows for easy access to the list of salient points in<br>an image ) . . . . .     | 559 |
| <a href="#">GistEstimator</a> . . . . .                                                                                     | 563 |
| <a href="#">GistEstimatorConfigurator</a> ( <a href="#">GistEstimator</a> configurator ) . . . . .                          | 566 |
| <a href="#">GpsDev</a> (Interface with a GPS unit ) . . . . .                                                               | 570 |
| <a href="#">GreenChannel</a> (A green color channel ) . . . . .                                                             | 576 |
| <a href="#">GreenChannelSurprise</a> (A green color channel ) . . . . .                                                     | 578 |
| <a href="#">Heading</a> (A simple class to represent 3D heading information ) . . . . .                                     | 581 |
| <a href="#">Histogram</a> (Histogram ) . . . . .                                                                            | 584 |
| <a href="#">Hmax</a> (Object recognition model inspired from Riesenhuber & Poggio's<br>HMAX model ) . . . . .               | 587 |
| <a href="#">HMR3300</a> (Class for interfacing with a Honeywell 3300 compass ) . . . . .                                    | 593 |
| <a href="#">HREntry</a> (Represents a <a href="#">HyperRectangle</a> entry in a list ) . . . . .                            | 596 |
| <a href="#">HueChannel</a> (A hue similarity channel ) . . . . .                                                            | 599 |
| <a href="#">HyperRectangle</a> (Represents a rectangle in n-dimensional space ) . . . . .                                   | 604 |
| <a href="#">IEEE1394grabber</a> (Definition and access functions for FireWire digital cam-<br>era frame capture ) . . . . . | 607 |
| <a href="#">Image&lt; T &gt;</a> (Generic image template class ) . . . . .                                                  | 611 |
| <a href="#">ImageCache&lt; T &gt;</a> (Base class for image caches that do computations on the<br>fly ) . . . . .           | 657 |
| <a href="#">ImageCacheAvg&lt; T &gt;</a> ( <a href="#">Image</a> cache to compute the running average ) . . . .             | 664 |

|                                                                                                                                         |     |
|-----------------------------------------------------------------------------------------------------------------------------------------|-----|
| <a href="#">ImageCacheMinMax&lt; T &gt;</a> (Image cache to compute a running min/max ) . . .                                           | 668 |
| <a href="#">ImageDisplay</a> (A generalized class for displaying images ) . . . . .                                                     | 671 |
| <a href="#">ImageMap</a> (3D image map created from the image at diff' orientations ) . . .                                             | 676 |
| <a href="#">ImageSet&lt; T &gt;</a> (This class implements a set of images, often used as a<br>dyadic pyramid ) . . . . .               | 677 |
| <a href="#">ImageSpring&lt; T &gt;</a> (This class is some sort of self organizing map used for<br>clustering ) . . . . .               | 684 |
| <a href="#">InferoTemporal</a> (InferoTemporal ) . . . . .                                                                              | 691 |
| <a href="#">InferoTemporalConfigurator</a> ( <a href="#">InferoTemporal</a> configurator ) . . . . .                                    | 695 |
| <a href="#">InformationChannel</a> (An information channel ) . . . . .                                                                  | 699 |
| <a href="#">InputFrameSeries</a> (A <a href="#">FrameSeries</a> with additional facilities for reading input<br>frames ) . . . . .      | 703 |
| <a href="#">IntensityChannel</a> (An intensity channel ) . . . . .                                                                      | 708 |
| <a href="#">IntensityChannelBeo</a> ( <a href="#">Beowulf</a> version of <a href="#">IntensityChannel</a> ) . . . . .                   | 710 |
| <a href="#">IntensityChannelSurprise</a> (An intensity channel ) . . . . .                                                              | 713 |
| <a href="#">Jet&lt; T &gt;</a> (This class specifies a Jet, i.e., array of feature values for one<br>location ) . . . . .               | 716 |
| <a href="#">JetSpec</a> (This class specifies the features, scales and indices represented in a<br><a href="#">Jet</a> ) . . . . .      | 723 |
| <a href="#">JunctionChannel</a> (A junction channel ) . . . . .                                                                         | 729 |
| <a href="#">JunctionChannelSurprise</a> (A junction channel ) . . . . .                                                                 | 733 |
| <a href="#">KalmanFilter</a> (Implementation of a 2nd order linear Kalman Filter ) . . . .                                              | 736 |
| <a href="#">KDTree</a> (A simple K-d tree implementation ) . . . . .                                                                    | 741 |
| <a href="#">KeyBoard</a> (Simple interface to a keyboard ) . . . . .                                                                    | 744 |
| <a href="#">KeypointMatch</a> (KeypointMatch ) . . . . .                                                                                | 746 |
| <a href="#">KnowledgeBase</a> (This class holds a knowledge base ) . . . . .                                                            | 749 |
| <a href="#">LaplacianPyrBuilder&lt; T &gt;</a> (Builds pyramids based on Laplacian filters ) . . .                                      | 753 |
| <a href="#">Lcd</a> (Lcd.H Interface to an LCD screen via serial port ) . . . . .                                                       | 755 |
| <a href="#">LeakyIntegrator</a> (A leaky integrator neuron, used by standard saliency map<br>(SM) ) . . . . .                           | 760 |
| <a href="#">LeakyIntFire</a> (A leaky integrate & fire neuron, used by standard winner-take-<br>all (WTA) ) . . . . .                   | 763 |
| <a href="#">LevelSpec</a> (A utility class for use with <a href="#">SingleChannel</a> ) . . . . .                                       | 766 |
| <a href="#">std::list&lt; T, Alloc &gt;</a> (The STL list container class. Use with <code>#include</code> ) . . .                       | 773 |
| <a href="#">LJunctionChannel</a> (An L junction channel that detects L junctions at different )                                         | 774 |
| <a href="#">LJunctionChannelSurprise</a> (An L junction channel that detects L junctions at<br>different orientations ) . . . . .       | 778 |
| <a href="#">LPTFoveator</a> (A class that does space-variant processing with log polar trans-<br>form ) . . . . .                       | 781 |
| <a href="#">Map</a> (Definition of Map class and access functions ) . . . . .                                                           | 787 |
| <a href="#">std::map&lt; Key, Data, Compare, Alloc &gt;</a> (The STL map container class. Use<br>with <code>#include</code> ) . . . . . | 791 |
| <a href="#">Matrix3</a> . . . . .                                                                                                       | 792 |
| <a href="#">MbariResultViewer</a> (Class to facilitate displaying and saving of intermediate<br>results ) . . . . .                     | 796 |
| <a href="#">Raster::MetaData</a> (Metadata information (typically obtained from GPS) ) . .                                              | 812 |

|                                                                                                                                                |     |
|------------------------------------------------------------------------------------------------------------------------------------------------|-----|
| <a href="#">MexBuf</a> (This streambuf is used to re-direct cout and cerr ) . . . . .                                                          | 817 |
| <a href="#">MexBufInit</a> (This class triggers the redirection of the standard streams at construction ) . . . . .                            | 818 |
| <a href="#">MexModelManager</a> (A special form of <a href="#">ModelManager</a> used in mex files ) . .                                        | 819 |
| <a href="#">MexReturn</a> (Class to help manage return values in mex files ) . . . . .                                                         | 824 |
| <a href="#">ModelComponent</a> (Base class for parameterized model components ) . . . .                                                        | 827 |
| <a href="#">ModelEyeHeadController</a> (Realistic Eye/Head controller ) . . . . .                                                              | 848 |
| <a href="#">ModelManager</a> (Manages a collection of <a href="#">ModelComponent</a> objects forming a model ) . . . . .                       | 854 |
| <a href="#">ModelOptionDef</a> (Specification of a known model option ) . . . . .                                                              | 868 |
| <a href="#">ModelParam&lt; T &gt;</a> (ModelParam<T> definitions ) . . . . .                                                                   | 871 |
| <a href="#">ModelParamBase</a> (ModelParamBase is a persistent <a href="#">ModelComponent</a> parameter base class ) . . . . .                 | 875 |
| <a href="#">ModelWrapperSaccadeController</a> (Realistic rhesus monkey saccade controller ) . . . . .                                          | 881 |
| <a href="#">Monkey2SaccadeController</a> (Realistic rhesus monkey saccade controller ) . .                                                     | 886 |
| <a href="#">MonkeyEyeHeadController</a> (Realistic Monkey Eye/Head controller ) . . . .                                                        | 891 |
| <a href="#">MonkeySaccadeController</a> (Realistic rhesus monkey saccade controller ) . .                                                      | 897 |
| <a href="#">MotionChannel</a> (A composite channel containing a set of direction channels )                                                    | 903 |
| <a href="#">MotionChannelBeo</a> ( <a href="#">Beowulf</a> version of <a href="#">MotionChannel</a> ) . . . . .                                | 907 |
| <a href="#">MotionChannelSurprise</a> (A composite channel containing a set of direction channels ) . . . . .                                  | 909 |
| <a href="#">MotTest</a> (Class for testing <a href="#">MotionChannel</a> ) . . . . .                                                           | 912 |
| <a href="#">MPEGStream</a> (MPEGStream is a wrapper class to read video frames from mpeg-1 ) . . . . .                                         | 914 |
| <a href="#">MultiColorBandChannel</a> (A composite channel containing a set of color-band channels ) . . . . .                                 | 919 |
| <a href="#">MultiColorBandChannelBeo</a> ( <a href="#">Beowulf</a> version of <a href="#">MultiColorBandChannel</a> ) .                        | 923 |
| <a href="#">MultiColorBandChannelSurprise</a> (A composite channel containing a set of color-band channels ) . . . . .                         | 925 |
| <a href="#">MultiConvolveChannel</a> (A generic multi-convolution channel ) . . . . .                                                          | 928 |
| <a href="#">MultiConvolveChannelSurprise</a> (A generic multi-convolution channel ) . . .                                                      | 932 |
| <a href="#">MultiDirectFeedChannel</a> (Multi Direct Feed Channel ) . . . . .                                                                  | 934 |
| <a href="#">std::multimap&lt; Key, Data, Compare, Alloc &gt;</a> (The STL multimap container class. Use with <code>#include</code> ) . . . . . | 938 |
| <a href="#">std::multiset&lt; Key, Compare, Alloc &gt;</a> (The STL multiset container class. Use with <code>#include</code> ) . . . . .       | 939 |
| <a href="#">MyBeoChipListener</a> (Our own little <a href="#">BeoChipListener</a> ) . . . . .                                                  | 940 |
| <a href="#">NewClass</a> (Definition and access functions for a new class ) . . . . .                                                          | 942 |
| <a href="#">ObjectBase</a> (ObjectBase represents the various object modules for recognition ) . . . . .                                       | 944 |
| <a href="#">OldFrameSeries</a> (Manages a series of input or output frames for vision.C::main() ) . . . . .                                    | 947 |
| <a href="#">Omni&lt; T &gt;</a> (Omni directional correction tool ) . . . . .                                                                  | 948 |
| <a href="#">Ontology</a> (Definition and access functions for Ontology ) . . . . .                                                             | 950 |
| <a href="#">OptionParser</a> (Manages the parsing of argc/argv command-line arguments ) .                                                      | 953 |



|                                                                                                                                              |      |
|----------------------------------------------------------------------------------------------------------------------------------------------|------|
| <a href="#">OptionSet</a> (Represents a set of available command-line options ) . . . . .                                                    | 958  |
| <a href="#">OrientationChannel</a> (A composite channel containing a set of orientation-tuned gabor channels ) . . . . .                     | 962  |
| <a href="#">OrientationChannelBeo</a> (Beowulf version of <a href="#">OrientationChannel</a> ) . . . . .                                     | 969  |
| <a href="#">OrientationChannelSurprise</a> (A composite channel containing a set of orientation-tuned gabor channels ) . . . . .             | 971  |
| <a href="#">OrientedPyrBuilder&lt; T &gt;</a> (Builds pyramid based on oriented Laplacian filters ) . . . . .                                | 974  |
| <a href="#">OutputFrameSeries</a> (A <a href="#">FrameSeries</a> with additional facilities for writing output frames ) . . . . .            | 976  |
| <a href="#">ParamMap</a> (Allows reading and writing of parameters to and from files or streams ) . . . . .                                  | 980  |
| <a href="#">ParamMap::Error</a> (Exception class thrown by <a href="#">ParamMap</a> functions in case of error ) . . . . .                   | 991  |
| <a href="#">PbmParser</a> (This class handles reading/parsing of PGM/PPM image files ) . . . . .                                             | 992  |
| <a href="#">PedestrianChannel</a> (A template-matching channel for detecting pedestrians ) . . . . .                                         | 996  |
| <a href="#">PedestrianChannelSurprise</a> (A template-matching channel for detecting pedestrians ) . . . . .                                 | 999  |
| <a href="#">PfmParser</a> (This class handles reading/parsing of PFM image files ) . . . . .                                                 | 1002 |
| <a href="#">PixRGB&lt; T &gt;</a> (Generic pixel class for color RGB pixels ) . . . . .                                                      | 1005 |
| <a href="#">PN03contrastChannel</a> (A Contrast channel (based on Parkhurst & Niebur 2003) ) . . . . .                                       | 1018 |
| <a href="#">PngParser</a> (This class handles reading/parsing of PNG image files ) . . . . .                                                 | 1021 |
| <a href="#">PngWriter</a> (Helper class for PNG-writing functions ) . . . . .                                                                | 1024 |
| <a href="#">Point&lt; T, N &gt;</a> . . . . .                                                                                                | 1025 |
| <a href="#">Point2D</a> (This is a basic class to encode 2D integer coordinates ) . . . . .                                                  | 1029 |
| <a href="#">Point2DT</a> (This is a fairly trivial (i, j, t) set representing a timestamped <a href="#">Point2D</a> ) . . . . .              | 1039 |
| <a href="#">promote_trait&lt; T1, T2 &gt;</a> (Promote from T1 and T2 to a type than can hold $T1 * T2$ ) . . . . .                          | 1042 |
| <a href="#">Property</a> (Definition of Property class and its functions ) . . . . .                                                         | 1043 |
| <a href="#">PropertyVectorSet</a> (Class that contains a set of property vectors for events ) . . . . .                                      | 1046 |
| <a href="#">PropHold</a> (This is a container class for the property at each hyper pixel for <a href="#">ContourNeuronCreate</a> ) . . . . . | 1049 |
| <a href="#">PsychoDisplay</a> (Class fo do various psychophysics displays ) . . . . .                                                        | 1054 |
| <a href="#">PsychoTask</a> (A summary struct to handle task settings ) . . . . .                                                             | 1068 |
| <a href="#">PvmMessage</a> (Definition and access functions for PVM messages ) . . . . .                                                     | 1070 |
| <a href="#">PyrBuilder&lt; T &gt;</a> (An interface class for creating dyadic pyramids from input images ) . . . . .                         | 1076 |
| <a href="#">PyrFoveator</a> (A class that does space-variant processing with dyadic pyramid ) . . . . .                                      | 1079 |
| <a href="#">RadioDecoder</a> (This class runs a thread which records servo position from sound card ) . . . . .                              | 1083 |
| <a href="#">Range&lt; T &gt;</a> (A class for managing min/max range information ) . . . . .                                                 | 1088 |
| <a href="#">readConfig</a> (Open a config file and parse it for data retrieval ) . . . . .                                                   | 1093 |
| <a href="#">readMatrix</a> . . . . .                                                                                                         | 1101 |
| <a href="#">Rectangle</a> (A basic rectangle class ) . . . . .                                                                               | 1102 |

|                                                                                                                                               |      |
|-----------------------------------------------------------------------------------------------------------------------------------------------|------|
| <a href="#">RedChannel</a> (A red color channel ) . . . . .                                                                                   | 1108 |
| <a href="#">RedChannelSurprise</a> (A red color channel ) . . . . .                                                                           | 1110 |
| <a href="#">RedGreenChannel</a> (A red-green double-opponent color channel ) . . . . .                                                        | 1113 |
| <a href="#">RedGreenChannelBeo</a> ( <a href="#">Beowulf</a> version of <a href="#">RedGreenChannel</a> ) . . . . .                           | 1115 |
| <a href="#">RedGreenChannelSurprise</a> (A red-green double-opponent color channel ) . . . . .                                                | 1118 |
| <a href="#">ReichardtPyrBuilder&lt; T &gt;</a> (This class implements Reichardt motion detec-<br>tions ) . . . . .                            | 1121 |
| <a href="#">Retina</a> (The retina base class ) . . . . .                                                                                     | 1124 |
| <a href="#">RetinaConfigurator</a> ( <a href="#">Retina</a> configurator ) . . . . .                                                          | 1130 |
| <a href="#">RetinaStd</a> (Standard <a href="#">Retina</a> implementation ) . . . . .                                                         | 1134 |
| <a href="#">RGBConvolveChannel</a> (A generic convolution channel using color filters ) . . . . .                                             | 1141 |
| <a href="#">RGBConvolvePyrBuilder&lt; T &gt;</a> (Builds pyramids based on arbitrary color<br>filters ) . . . . .                             | 1144 |
| <a href="#">SaccadeController</a> (A base class to implement saccade controllers ) . . . . .                                                  | 1147 |
| <a href="#">SaccadeControllerConfigurator</a> ( <a href="#">SaccadeController</a> configurator ) . . . . .                                    | 1162 |
| <a href="#">SafeCopyHelper&lt; T, isTrivial &gt;</a> (Helper template struct for pointer version<br>of <a href="#">safecopy()</a> ) . . . . . | 1166 |
| <a href="#">SafeCopyHelper&lt; T, true &gt;</a> (Specialized implementation for trivial types ) . . . . .                                     | 1167 |
| <a href="#">SaliencyMap</a> (The saliency map base class ) . . . . .                                                                          | 1168 |
| <a href="#">SaliencyMapConfigurator</a> ( <a href="#">SaliencyMap</a> configurator ) . . . . .                                                | 1176 |
| <a href="#">SaliencyMapFast</a> (The fast saliency map: a 2D array of floats ) . . . . .                                                      | 1179 |
| <a href="#">SaliencyMapStd</a> (The standard saliency map: a 2D array of <a href="#">LeakyIntegrator</a><br>neurons ) . . . . .               | 1186 |
| <a href="#">SaliencyMapTrivial</a> (The trivial saliency map: a 2D array of floats ) . . . . .                                                | 1193 |
| <a href="#">ScaleSpace</a> (ScaleSpace ) . . . . .                                                                                            | 1198 |
| <a href="#">ScorrChannel</a> (A spatial correlation channel ) . . . . .                                                                       | 1201 |
| <a href="#">Serial</a> (Interface to a serial port ) . . . . .                                                                                | 1205 |
| <a href="#">std::set&lt; Key, Compare, Alloc &gt;</a> (The STL set container class. Use with<br><a href="#">#include</a> ) . . . . .          | 1215 |
| <a href="#">ShapeEstimator</a> (Everything realted to the shape estimation procedure ) . . . . .                                              | 1216 |
| <a href="#">SharedPtr&lt; T &gt;</a> (A smart pointer with reference counted copy semantics ) . . . . .                                       | 1225 |
| <a href="#">SimpleEyeHeadSaccadeController</a> (Simple eye/head controller ) . . . . .                                                        | 1231 |
| <a href="#">SimulationViewer</a> (This class provides an abstract interface for a simulation<br>viewer ) . . . . .                            | 1235 |
| <a href="#">SimulationViewerCompress</a> (Do multi-foveated saliency-based image com-<br>pression ) . . . . .                                 | 1242 |
| <a href="#">SimulationViewerConfigurator</a> ( <a href="#">SimulationViewer</a> configurator ) . . . . .                                      | 1258 |
| <a href="#">SimulationViewerEyeMvt</a> (Measure salience at human eye positions ) . . . . .                                                   | 1262 |
| <a href="#">SimulationViewerEyeMvt2</a> (Measure salience at human eye positions ) . . . . .                                                  | 1272 |
| <a href="#">SimulationViewerStd</a> (This class provides a standard set of visualization<br>tools ) . . . . .                                 | 1283 |
| <a href="#">SingleChannel</a> (SingleChannel represents a single stream of processing ) . . . . .                                             | 1297 |
| <a href="#">SingleChannelBeo</a> (Wrapper to run a <a href="#">SingleChannel</a> on a distant CPU ) . . . . .                                 | 1323 |
| <a href="#">SingleChannelBeoServer</a> (Server to work with <a href="#">SingleChannelBeo</a> ) . . . . .                                      | 1327 |
| <a href="#">SingleChannelSurprise</a> (SingleChannelSurprise is a Surprise extension to<br><a href="#">SingleChannel</a> ) . . . . .          | 1331 |

|                                                                                                                                        |      |
|----------------------------------------------------------------------------------------------------------------------------------------|------|
| <a href="#">SkinHueChannel</a> (A channel for detecting skin color ) . . . . .                                                         | 1337 |
| <a href="#">std::slist&lt; T, Alloc &gt;</a> (The STL slist container class. Use with <code>#include</code> ) . . . . .                | 1339 |
| <a href="#">SOBlueYellowChannel</a> (Single Opponent <a href="#">BlueYellowChannel</a> member definitions: ) . . . . .                 | 1340 |
| <a href="#">SOBlueYellowChannelSurprise</a> (Single Opponent <a href="#">BlueYellowChannelSurprise</a> member definitions: ) . . . . . | 1342 |
| <a href="#">SockServ</a> (This is a multi-socket connection server ) . . . . .                                                         | 1345 |
| <a href="#">SOColorChannel</a> . . . . .                                                                                               | 1352 |
| <a href="#">SOColorChannelSurprise</a> (A single opponent color channel class that combines r/g, g/r, b/y, etc ) . . . . .             | 1359 |
| <a href="#">SOGreenRedChannel</a> (Single Opponent <a href="#">GreenRedChannel</a> member definitions: ) . . . . .                     | 1364 |
| <a href="#">SOGreenRedChannelSurprise</a> (Single Opponent <a href="#">GreenRedChannelSurprise</a> member definitions: ) . . . . .     | 1366 |
| <a href="#">SORedGreenChannel</a> (Single Opponent <a href="#">RedGreenChannel</a> member definitions: ) . . . . .                     | 1369 |
| <a href="#">SORedGreenChannelSurprise</a> (Single Opponent <a href="#">RedGreenChannelSurprise</a> member definitions: ) . . . . .     | 1371 |
| <a href="#">SortedLimitedList&lt; T &gt;</a> (Maintains a sorted list with a limited number of elements ) . . . . .                    | 1374 |
| <a href="#">SoxChannel</a> (A short-range orientation interactions channel ) . . . . .                                                 | 1378 |
| <a href="#">SOYellowBlueChannel</a> (Single Opponent <a href="#">YellowBlueChannel</a> member definitions: ) . . . . .                 | 1384 |
| <a href="#">SOYellowBlueChannelSurprise</a> (Single Opponent <a href="#">YellowBlueChannelSurprise</a> member definitions: ) . . . . . | 1386 |
| <a href="#">SpeedSensor</a> (This class allows to use a PS/2 mouse as a speed sensor ) . . . . .                                       | 1389 |
| <a href="#">SSC</a> (Interface to a serial-servo-controller ) . . . . .                                                                | 1390 |
| <a href="#">Staircase</a> (A staircase procedure to measure psychophysical thresholds ) . . . . .                                      | 1396 |
| <a href="#">StdBrain</a> (A subclass of <a href="#">Brain</a> that automatically sets up the standard modules ) . . . . .              | 1402 |
| <a href="#">StereoChannel</a> (A composite channel containing a set of Disparity channels ) . . . . .                                  | 1405 |
| <a href="#">StraightLine2D</a> (A straight line in the 2D plane ) . . . . .                                                            | 1411 |
| <a href="#">std::string</a> (The STL string class. Use with <code>#include</code> ) . . . . .                                          | 1416 |
| <a href="#">SurpriseImage&lt; T &gt;</a> (A 2D array of <a href="#">SurpriseModel</a> objects ) . . . . .                              | 1417 |
| <a href="#">SurpriseMap&lt; T &gt;</a> (A <a href="#">Surprise Map</a> ) . . . . .                                                     | 1422 |
| <a href="#">SurpriseModel</a> (A local (single-point) surprise model ) . . . . .                                                       | 1425 |
| <a href="#">SurpriseModelSG</a> (A single-Gaussian <a href="#">SurpriseModel</a> ) . . . . .                                           | 1431 |
| <a href="#">SurpriseModelSP</a> (A single-Poisson/Gamma <a href="#">SurpriseModel</a> ) . . . . .                                      | 1436 |
| <a href="#">TaskRelevanceMap</a> (The task-relevance map base class ) . . . . .                                                        | 1441 |
| <a href="#">TaskRelevanceMapConfigurator</a> ( <a href="#">TaskRelevanceMap</a> configurator ) . . . . .                               | 1447 |
| <a href="#">TaskRelevanceMapKillN</a> (A KillN task-relevance map ) . . . . .                                                          | 1451 |
| <a href="#">TaskRelevanceMapKillStatic</a> (A killStatic task-relevance map ) . . . . .                                                | 1456 |
| <a href="#">TaskRelevanceMapStd</a> (The standard task-relevance map ) . . . . .                                                       | 1462 |
| <a href="#">TcorrChannel</a> (A temporal correlation channel ) . . . . .                                                               | 1468 |
| <a href="#">TCPcliServ</a> (A client/server to receive/send <a href="#">TCPmessage</a> ) . . . . .                                     | 1472 |
| <a href="#">TCPcommunicator</a> (This class handles passing TCPmessages in and out ) . . . . .                                         | 1476 |

|                                                                                                                                                                            |      |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------|
| <a href="#">TCPmessage</a> (This class defines a modular message to be passed over TCP connections ) . . . . .                                                             | 1485 |
| <a href="#">TemplateMatchChannel</a> (Convolve each level of its pyramid with a feature template ) . . . . .                                                               | 1494 |
| <a href="#">TemplateMatchChannelSurprise</a> (Convolve each level of its pyramid with a feature template ) . . . . .                                                       | 1497 |
| <a href="#">TemplateMatchPyrBuilder</a> (A <a href="#">PyrBuilder</a> that does template matching ) . . .                                                                  | 1500 |
| <a href="#">TestSuite</a> (TestSuite manages a set of tests ) . . . . .                                                                                                    | 1502 |
| <a href="#">ThresholdFrictionSaccadeController</a> (Threshold-limited friction-damped controller ) . . . . .                                                               | 1508 |
| <a href="#">ThresholdSaccadeController</a> (Threshold-limited controller that eliminates small or unreliable shifts ) . . . . .                                            | 1513 |
| <a href="#">Timer</a> (This class implements a real-time timer ) . . . . .                                                                                                 | 1519 |
| <a href="#">TJunctionChannel</a> (A T junction channel that detects T junctions at different )                                                                             | 1523 |
| <a href="#">TJunctionChannelSurprise</a> (A T junction channel that detects T junctions at different orientations ) . . . . .                                              | 1527 |
| <a href="#">Token</a> (Public class that contains information for a visual token for tracking )                                                                            | 1530 |
| <a href="#">ToolkitInit</a> (A helper class for program-wide initialization ) . . . . .                                                                                    | 1535 |
| <a href="#">TrivialSaccadeController</a> (Trivial saccade controller ) . . . . .                                                                                           | 1537 |
| <a href="#">TWM</a> (Definition and access functions for TWM ) . . . . .                                                                                                   | 1540 |
| <a href="#">type_if&lt; T, B &gt;</a> (Makes a <i>type</i> a typedef for <i>T</i> only if <i>B</i> is true ) . . . . .                                                     | 1544 |
| <a href="#">type_with_N_bits&lt; T, N &gt;</a> (A compile-time check that <i>T</i> has <i>N</i> bits ) . . . .                                                             | 1545 |
| <a href="#">TypeTraits&lt; T &gt;</a> (The generic TypeTraits implementation ) . . . . .                                                                                   | 1546 |
| <a href="#">TypeTraits&lt; PixRGB&lt; T &gt; &gt;</a> (Specialize <a href="#">TypeTraits</a> to indicate that <a href="#">Pix-<br/>RGB</a> is a "trivial" type ) . . . . . | 1547 |
| <a href="#">V4Lgrabber</a> (Definition and access functions for video4linux frame grabber )                                                                                | 1548 |
| <a href="#">V4LgrabberLight</a> (Wrapper class fro <a href="#">V4Lgrabber</a> that restores the original interface ) . . . . .                                             | 1552 |
| <a href="#">V4LPalette</a> (A V4L palette entry ) . . . . .                                                                                                                | 1554 |
| <a href="#">VarianceChannel</a> (A variance channel, computes the variance in an image patch ) . . . . .                                                                   | 1556 |
| <a href="#">VCC4</a> (Interface with the pan-tilt mechanism of a Canon VC-C4 camera ) . .                                                                                  | 1559 |
| <a href="#">std::vector&lt; T, Alloc &gt;</a> (The STL vector container class. Use with <code>#include</code> ) . . . . .                                                  | 1571 |
| <a href="#">Vector2D</a> (A vector in the 2D plane ) . . . . .                                                                                                             | 1572 |
| <a href="#">Vector3</a> . . . . .                                                                                                                                          | 1581 |
| <a href="#">VisBufSaccadeController</a> (VisBuf saccade controller ) . . . . .                                                                                             | 1588 |
| <a href="#">VisualBuffer</a> (An integrative visual buffer ) . . . . .                                                                                                     | 1592 |
| <a href="#">VisualCortex</a> (The Visual Cortex Class ) . . . . .                                                                                                          | 1601 |
| <a href="#">VisualCortexBeo</a> (This is a derivation of <a href="#">VisualCortex</a> that uses <a href="#">Single-<br/>ChannelBeo</a> channels ) . . . . .                | 1618 |
| <a href="#">VisualCortexConfigurator</a> ( <a href="#">VisualCortex</a> configurator ) . . . . .                                                                           | 1622 |
| <a href="#">VisualCortexEyeMvt</a> (The VisualCortexEyeMvt Class ) . . . . .                                                                                               | 1626 |
| <a href="#">VisualCortexSurprise</a> (This is a derivation of <a href="#">VisualCortex</a> using <a href="#">Single-<br/>ChannelSurprise</a> channels ) . . . . .          | 1633 |

|                                                                                                                           |      |
|---------------------------------------------------------------------------------------------------------------------------|------|
| <a href="#">VisualEvent</a> (Stores a visual event as a sequence of tokens with additional information ) . . . . .        | 1636 |
| <a href="#">VisualEventSet</a> (Set of visual events to which a set of tokens can be matched ) . . . . .                  | 1643 |
| <a href="#">VisualObject</a> (Describes name and attributes of a visual object ) . . . . .                                | 1650 |
| <a href="#">VisualObjectDB</a> (VisualObjectDatabase ) . . . . .                                                          | 1657 |
| <a href="#">VisualObjectMatch</a> (VisualObjectMatch ) . . . . .                                                          | 1661 |
| <a href="#">WinnerTakeAll</a> (The winner-take-all base class ) . . . . .                                                 | 1665 |
| <a href="#">WinnerTakeAllConfigurator</a> ( <a href="#">WinnerTakeAll</a> configurator ) . . . . .                        | 1672 |
| <a href="#">WinnerTakeAllDummy</a> (A dummy winner-take-all ) . . . . .                                                   | 1676 |
| <a href="#">WinnerTakeAllFast</a> (A fast winner-take-all ) . . . . .                                                     | 1681 |
| <a href="#">WinnerTakeAllStd</a> (The standard winner-take-all ) . . . . .                                                | 1686 |
| <a href="#">WTAwinner</a> (This is a an open class representing a WTA winner (covert attention shift) ) . . . . .         | 1691 |
| <a href="#">XJunctionChannel</a> (A X junction channel that detects crosses at different ) . . . . .                      | 1696 |
| <a href="#">XJunctionChannelSurprise</a> (A X junction channel that detects crosses at different orientations ) . . . . . | 1700 |
| <a href="#">XWindow</a> (A simple X-window to display images ) . . . . .                                                  | 1703 |
| <a href="#">XWinManaged</a> (A window class with active window management ) . . . . .                                     | 1708 |
| <a href="#">YellowChannel</a> (A yellow double-opponent color channel ) . . . . .                                         | 1715 |
| <a href="#">YellowChannelSurprise</a> (A yellow double-opponent color channel ) . . . . .                                 | 1717 |



## Chapter 6

# iLab Neuromorphic Vision Toolkit File Index

### 6.1 iLab Neuromorphic Vision Toolkit File List

Here is a list of all documented files with brief descriptions:

|                                        |      |
|----------------------------------------|------|
| <b>Agent.C</b>                         | ??   |
| <b>Agent.H</b>                         | ??   |
| <a href="#">ArrayData.H</a>            | 1721 |
| <a href="#">atomic.H</a>               | 1726 |
| <a href="#">AttentionGuidanceMap.C</a> | 1728 |
| <a href="#">AttentionGuidanceMap.H</a> | 1729 |
| <a href="#">AudioGrabber.C</a>         | 1730 |
| <a href="#">AudioGrabber.H</a>         | 1731 |
| <a href="#">AudioMixer.C</a>           | 1732 |
| <a href="#">AudioMixer.H</a>           | 1733 |
| <a href="#">beobot-defs.H</a>          | 1734 |
| <a href="#">beobot-master.C</a>        | 1736 |
| <a href="#">beobot-slave.C</a>         | 1740 |
| <a href="#">Beobot.C</a>               | 1743 |
| <a href="#">Beobot.H</a>               | 1744 |
| <a href="#">BeobotAction.H</a>         | 1745 |
| <a href="#">BeobotEffectors.C</a>      | 1746 |
| <a href="#">BeobotEffectors.H</a>      | 1747 |
| <a href="#">BeobotMemory.C</a>         | 1748 |
| <a href="#">BeobotMemory.H</a>         | 1749 |
| <a href="#">BeobotSensation.C</a>      | 1751 |
| <a href="#">BeobotSensation.H</a>      | 1752 |
| <a href="#">BeobotSensors.C</a>        | 1754 |

|                                |           |
|--------------------------------|-----------|
| BeobotSensors.H                | 1755      |
| BeobotVisualCortex.C           | 1756      |
| BeobotVisualCortex.H           | 1757      |
| beochannel-server.C            | 1759      |
| BeoChip.C                      | 1760      |
| BeoChip.H                      | 1761      |
| beograb-master.C               | 1763      |
| beograb.C                      | 1765      |
| BeoSub-defs.H                  | 1767      |
| BeoSub.C                       | 1771      |
| BeoSub.H                       | 1773      |
| BeoSubAction.C                 | 1776      |
| BeoSubAction.H                 | 1777      |
| BeoSubDepthSensor.C            | 1778      |
| BeoSubDepthSensor.H            | 1779      |
| <b>BeoSubMotor.C</b>           | <b>??</b> |
| <b>BeoSubMotor.H</b>           | <b>??</b> |
| Beowulf.C                      | 1780      |
| Beowulf.H                      | 1781      |
| BitObject.C                    | 1784      |
| BitObject.H                    | 1786      |
| BitObjectDrawModes.H           | 1788      |
| blobProp.H                     | 1790      |
| BlurFoveator.C                 | 1791      |
| BlurFoveator.H                 | 1792      |
| bmcvfigs.C                     | 1794      |
| BPneuron.C                     | 1796      |
| BPneuron.H                     | 1797      |
| BPnnnet.C                      | 1798      |
| BPnnnet.H                      | 1799      |
| Brain.C                        | 1801      |
| Brain.H                        | 1803      |
| calibrateCamera.C              | 1806      |
| calibrateFunctions.C           | 1807      |
| calibrateFunctions.H           | 1808      |
| calibrateVCC4.C                | 1809      |
| cameraConf.C                   | 1810      |
| cameraConf.H                   | 1811      |
| CameraControl.C                | 1812      |
| CameraControl.H                | 1813      |
| <b>CameraCoordTransforms.C</b> | <b>??</b> |
| <b>CameraCoordTransforms.H</b> | <b>??</b> |
| carConf.C                      | 1815      |
| carConf.H                      | 1816      |
| carcontrol.C                   | 1817      |
| carcontrol.H                   | 1818      |



|                                          |      |
|------------------------------------------|------|
| <a href="#">cascadeHold.H</a>            | 1820 |
| <a href="#">ChannelBase.C</a>            | 1821 |
| <a href="#">ChannelBase.H</a>            | 1822 |
| <a href="#">Channels.C</a>               | 1825 |
| <a href="#">Channels.H</a>               | 1828 |
| <a href="#">ChannelsBeo.C</a>            | 1834 |
| <a href="#">ChannelsBeo.H</a>            | 1835 |
| <b>ChannelsSurprise.C</b>                | ??   |
| <a href="#">ChannelsSurprise.H</a>       | 1837 |
| <a href="#">CheckedIterator.H</a>        | 1840 |
| <a href="#">CINNIC.C</a>                 | 1841 |
| <a href="#">CINNIC.H</a>                 | 1842 |
| <a href="#">CINNICanova.C</a>            | 1844 |
| <a href="#">CINNICinteract.C</a>         | 1848 |
| <a href="#">CINNICstats.C</a>            | 1851 |
| <a href="#">CINNICstatsRun.C</a>         | 1854 |
| <a href="#">CINNICstatsRun.H</a>         | 1855 |
| <a href="#">CINNICtest.C</a>             | 1856 |
| <a href="#">ColorComputeTypes.H</a>      | 1859 |
| <a href="#">colorDefs.H</a>              | 1861 |
| <b>ColorSignature.C</b>                  | ??   |
| <b>ColorSignature.H</b>                  | ??   |
| <a href="#">ComplexChannel.C</a>         | 1862 |
| <a href="#">ComplexChannel.H</a>         | 1863 |
| <a href="#">COMPort.C</a>                | 1865 |
| <a href="#">COMPort.H</a>                | 1866 |
| <a href="#">computeClampedDiffMean.C</a> | 1868 |
| <b>condensationTrack.C</b>               | ??   |
| <a href="#">Contour.C</a>                | 1869 |
| <b>ContourChannel.C</b>                  | ??   |
| <b>ContourChannel.H</b>                  | ??   |
| <a href="#">contourDefine.H</a>          | 1872 |
| <a href="#">contourNeuron.C</a>          | 1873 |
| <a href="#">contourNeuron.H</a>          | 1875 |
| <a href="#">contourNeuronProp.C</a>      | 1877 |
| <a href="#">contourNeuronProp.H</a>      | 1878 |
| <a href="#">contourRun.C</a>             | 1879 |
| <a href="#">contourRun.H</a>             | 1880 |
| <b>coord.C</b>                           | ??   |
| <b>corrcoef.C</b>                        | ??   |
| <a href="#">covEstimate.C</a>            | 1882 |
| <a href="#">covEstimate.H</a>            | 1883 |
| <a href="#">covHolder.H</a>              | 1884 |
| <b>CpuTimer.H</b>                        | ??   |
| <a href="#">create_thumbnails.C</a>      | 1885 |
| <b>dataFrameGrab.C</b>                   | ??   |

|                                        |      |
|----------------------------------------|------|
| <b>density.C</b>                       | ??   |
| <b>density.H</b>                       | ??   |
| <a href="#">Dims.H</a>                 | 1886 |
| <a href="#">DirCTest.C</a>             | 1888 |
| <a href="#">DirectFeedChannel.C</a>    | 1890 |
| <a href="#">DirectFeedChannel.H</a>    | 1891 |
| <a href="#">DirTest.C</a>              | 1893 |
| <a href="#">DirTest.H</a>              | 1895 |
| <a href="#">dummySTL.H</a>             | 1896 |
| <a href="#">eyemap.C</a>               | 1897 |
| <a href="#">ezvision.C</a>             | 1899 |
| <b>faceRecognition.C</b>               | ??   |
| <a href="#">fade_image.C</a>           | 1900 |
| <a href="#">fancynorm.C</a>            | 1901 |
| <a href="#">fancynorm.H</a>            | 1903 |
| <a href="#">fCV_STM.C</a>              | 1908 |
| <a href="#">fCV_STM.H</a>              | 1909 |
| <a href="#">featureClusterVision.C</a> | 1910 |
| <b>featureClusterVision.H</b>          | ??   |
| <a href="#">FFN.C</a>                  | 1911 |
| <a href="#">FFN.H</a>                  | 1912 |
| <b>FFTWWrapper.C</b>                   | ??   |
| <b>FFTWWrapper.H</b>                   | ??   |
| <b>findColorIndex.C</b>                | ??   |
| <b>findColorIndex.H</b>                | ??   |
| <a href="#">FOEstimator.C</a>          | 1913 |
| <a href="#">FOEstimator.H</a>          | 1914 |
| <b>font.c</b>                          | ??   |
| <a href="#">font2c.C</a>               | 1915 |
| <a href="#">Foveator.C</a>             | 1917 |
| <a href="#">Foveator.H</a>             | 1918 |
| <a href="#">fpe.C</a>                  | 1919 |
| <a href="#">fpe.H</a>                  | 1922 |
| <a href="#">FrameGrabber.C</a>         | 1925 |
| <a href="#">FrameGrabber.H</a>         | 1926 |
| <a href="#">FrameGrabberMode.H</a>     | 1928 |
| <a href="#">FrameRange.H</a>           | 1930 |
| <a href="#">FrameSeries.C</a>          | 1931 |
| <a href="#">FrameSeries.H</a>          | 1932 |
| <a href="#">fzvision.C</a>             | 1935 |
| <a href="#">gaborElement.C</a>         | 1936 |
| <a href="#">gaborElement.H</a>         | 1937 |
| <a href="#">GaborPyramid.C</a>         | 1938 |
| <a href="#">GaborPyramid.H</a>         | 1940 |
| <a href="#">GACHromosome.C</a>         | 1942 |
| <a href="#">GACHromosome.H</a>         | 1944 |

|                             |           |
|-----------------------------|-----------|
| GAMPIvision.C               | 1946      |
| GAPopulation.C              | 1948      |
| GAPopulation.H              | 1949      |
| Geometry2D.C                | 1951      |
| Geometry2D.H                | 1953      |
| getSaliency.C               | 1955      |
| getSaliency.H               | 1957      |
| getShapeEstimator.C         | 1958      |
| GistEstimator.C             | 1961      |
| GistEstimator.H             | 1964      |
| GistEstimatorConfigurator.C | 1965      |
| GistEstimatorConfigurator.H | 1966      |
| <b>gistParams.H</b>         | <b>??</b> |
| GPS.C                       | 1967      |
| GPS.H                       | 1968      |
| grab-server.C               | 1970      |
| Graph.C                     | 1972      |
| Graph.H                     | 1973      |
| gtk-SoxChannel.C            | 1975      |
| Gyro.C                      | 1979      |
| Gyro.H                      | 1980      |
| Histogram.C                 | 1981      |
| Histogram.H                 | 1982      |
| Hmax.C                      | 1983      |
| Hmax.H                      | 1984      |
| HMR3300.C                   | 1985      |
| HMR3300.H                   | 1986      |
| <b>hysteresis.C</b>         | <b>??</b> |
| <b>hysteresis.H</b>         | <b>??</b> |
| IEEE1394grabber.C           | 1988      |
| IEEE1394grabber.H           | 1989      |
| Image.C                     | 1992      |
| Image.H                     | 1993      |
| Image_All.H                 | 1994      |
| Image_ColorOps.C            | 1997      |
| Image_ColorOps.H            | 2016      |
| <b>Image_DrawOps.C</b>      | <b>??</b> |
| <b>Image_DrawOps.H</b>      | <b>??</b> |
| Image_FilterOps.C           | 2035      |
| Image_FilterOps.H           | 2049      |
| Image_IO.C                  | 2069      |
| Image_IO.H                  | 2072      |
| Image_MathOps.C             | 2075      |
| Image_MathOps.H             | 2094      |
| Image_MonoOnly.C            | 2116      |
| Image_Omni.C                | 2117      |

|                                    |           |
|------------------------------------|-----------|
| Image_Omni.H                       | 2121      |
| Image_ShapeOps.C                   | 2125      |
| Image_ShapeOps.H                   | 2128      |
| Image_Transforms.C                 | 2138      |
| Image_Transforms.H                 | 2147      |
| ImageCache.C                       | 2158      |
| ImageCache.H                       | 2160      |
| ImageConvertQt.C                   | 2162      |
| ImageConvertQt.H                   | 2164      |
| ImageDisplay.C                     | 2166      |
| ImageDisplay.H                     | 2167      |
| ImageDisplayTypes.H                | 2168      |
| ImageMap.H                         | 2170      |
| ImageSet.C                         | 2171      |
| ImageSet.H                         | 2172      |
| ImageSet_Ops.C                     | 2175      |
| ImageSet_Ops.H                     | 2187      |
| ImageSpring.C                      | 2200      |
| ImageSpring.H                      | 2201      |
| InferoTemporal.C                   | 2203      |
| InferoTemporal.H                   | 2204      |
| InferoTemporalConfigurator.C       | 2205      |
| InferoTemporalConfigurator.H       | 2206      |
| InstantiateAll.H                   | 2207      |
| IOTypes.H                          | 2208      |
| <b>IRoom.C</b>                     | <b>??</b> |
| <b>IRoom.H</b>                     | <b>??</b> |
| <b>IRoom_AgentMinder.C</b>         | <b>??</b> |
| <b>IRoom_AgentMinder.H</b>         | <b>??</b> |
| <b>IRoom_CameraManager.C</b>       | <b>??</b> |
| <b>IRoom_CameraManager.H</b>       | <b>??</b> |
| <b>IRoom_ExampleModule.C</b>       | <b>??</b> |
| <b>IRoom_ExampleModule.H</b>       | <b>??</b> |
| <b>IRoom_PursuitTracker.C</b>      | <b>??</b> |
| <b>IRoom_PursuitTracker.H</b>      | <b>??</b> |
| <b>IRoom_SignatureRepository.C</b> | <b>??</b> |
| <b>IRoom_SignatureRepository.H</b> | <b>??</b> |
| <b>IRoom_UserInterface.C</b>       | <b>??</b> |
| <b>IRoom_UserInterface.H</b>       | <b>??</b> |
| <b>IRoomAction.C</b>               | <b>??</b> |
| <b>IRoomAction.H</b>               | <b>??</b> |
| <b>IRoomAgentInfo.C</b>            | <b>??</b> |
| <b>IRoomAgentInfo.H</b>            | <b>??</b> |
| <b>IRoomCallJavaMsg.C</b>          | <b>??</b> |
| <b>IRoomCallJavaMsg.H</b>          | <b>??</b> |
| <b>IRoomCameraInfo.C</b>           | <b>??</b> |

|                                |      |
|--------------------------------|------|
| <b>IRoomCameraInfo.H</b>       | ??   |
| <b>IRoomCommandLine.C</b>      | ??   |
| <b>IRoomCommandLine.H</b>      | ??   |
| <b>IRoomFindSignalCamera.C</b> | ??   |
| <b>IRoomFindSignalCamera.H</b> | ??   |
| <b>IRoomHandoff.C</b>          | ??   |
| <b>IRoomHandoff.H</b>          | ??   |
| <b>IRoomMoveSignal.C</b>       | ??   |
| <b>IRoomMoveSignal.H</b>       | ??   |
| <b>IRoomResource.C</b>         | ??   |
| <b>IRoomResource.H</b>         | ??   |
| <b>IRoomResourceReport.H</b>   | ??   |
| <b>IRoomSaccade.C</b>          | ??   |
| <b>IRoomSaccade.H</b>          | ??   |
| Jet.C                          | 2210 |
| Jet.H                          | 2212 |
| KalmanFilter.C                 | 2215 |
| KalmanFilter.H                 | 2216 |
| KDTree.C                       | 2217 |
| KDTree.H                       | 2218 |
| KeyBoard.C                     | 2220 |
| KeyBoard.H                     | 2221 |
| Keypoint.C                     | 2223 |
| Keypoint.H                     | 2224 |
| KnowledgeBase.C                | 2225 |
| KnowledgeBase.H                | 2226 |
| Landmark_Graph.C               | 2227 |
| Landmark_Graph.H               | 2228 |
| lcd.C                          | 2230 |
| lcd.H                          | 2231 |
| lcdecho.C                      | 2232 |
| LeakyIntegrator.H              | 2233 |
| LeakyIntFire.H                 | 2234 |
| learnvision.C                  | 2235 |
| LevelSpec.C                    | 2237 |
| LevelSpec.H                    | 2238 |
| log.C                          | 2241 |
| log.H                          | 2243 |
| LPTFoveator.C                  | 2253 |
| LPTFoveator.H                  | 2254 |
| lumin.C                        | 2256 |
| <b>mainpage.dxy</b>            | ??   |
| Make_map.C                     | 2259 |
| Make_map.H                     | 2260 |
| make_target_image.C            | 2261 |
| make_test_movie.C              | 2262 |

|                                            |      |
|--------------------------------------------|------|
| <b>Matrix3.C</b>                           | ??   |
| <b>Matrix3.H</b>                           | ??   |
| <a href="#">mbariFunctions.C</a>           | 2263 |
| <a href="#">mbariFunctions.H</a>           | 2266 |
| <a href="#">MbariResultViewer.C</a>        | 2269 |
| <a href="#">MbariResultViewer.H</a>        | 2271 |
| <a href="#">mexConverts.C</a>              | 2272 |
| <a href="#">mexConverts.H</a>              | 2278 |
| <a href="#">mexDummy.C</a>                 | 2284 |
| <a href="#">mexLog.C</a>                   | 2285 |
| <a href="#">mexLog.H</a>                   | 2287 |
| <a href="#">MexModelManager.C</a>          | 2290 |
| <a href="#">MexModelManager.H</a>          | 2291 |
| <a href="#">mexSaliency.C</a>              | 2292 |
| <b>mexSaliencySkin.C</b>                   | ??   |
| <a href="#">mexTest.C</a>                  | 2295 |
| <a href="#">mexTestDirectFeedChannel.C</a> | 2297 |
| <a href="#">mexTestReichardt.C</a>         | 2299 |
| <a href="#">mexVision.C</a>                | 2301 |
| <a href="#">mmx-sse.C</a>                  | 2304 |
| <a href="#">mmx-sse.H</a>                  | 2305 |
| <b>modelbuilding.dxy</b>                   | ??   |
| <a href="#">ModelComponent.C</a>           | 2307 |
| <a href="#">ModelComponent.H</a>           | 2308 |
| <b>ModelFace.H</b>                         | ??   |
| <a href="#">ModelManager.C</a>             | 2311 |
| <a href="#">ModelManager.H</a>             | 2313 |
| <a href="#">ModelManagerControl.ui.h</a>   | 2316 |
| <a href="#">ModelManagerDialog.ui.h</a>    | 2317 |
| <a href="#">ModelOptionAliases.H</a>       | 2318 |
| <a href="#">ModelOptionDefs.C</a>          | 2319 |
| <a href="#">ModelOptionDefs.H</a>          | 2321 |
| <a href="#">ModelParam.C</a>               | 2329 |
| <a href="#">ModelParam.H</a>               | 2332 |
| <a href="#">MotCTest.C</a>                 | 2333 |
| <a href="#">MotTest.C</a>                  | 2335 |
| <a href="#">MotTest.H</a>                  | 2337 |
| <a href="#">MPEGStream.C</a>               | 2339 |
| <a href="#">MPEGStream.H</a>               | 2343 |
| <a href="#">mpgsize.C</a>                  | 2345 |
| <a href="#">MPIopenvision.C</a>            | 2346 |
| <a href="#">MPIopenvision2.C</a>           | 2348 |
| <a href="#">MPIopenvision3.C</a>           | 2351 |
| <a href="#">MrGabor.C</a>                  | 2354 |
| <a href="#">MrGabor2.C</a>                 | 2359 |
| <a href="#">nerd-cam-wan.C</a>             | 2364 |

|                        |      |
|------------------------|------|
| nerd-camTCP2-master.C  | 2366 |
| nerd-camTCP2.C         | 2371 |
| NewClass.C             | 2374 |
| NewClass.H             | 2375 |
| NPclassify-test.C      | 2376 |
| NPclassify.C           | 2378 |
| NPclassify.H           | 2379 |
| NPclassify2.C          | 2380 |
| NPclassify2.H          | 2381 |
| <b>nrutil.h</b>        | ??   |
| <b>ObjectBase.C</b>    | ??   |
| <b>ObjectBase.H</b>    | ??   |
| Omni.C                 | 2382 |
| Omni.H                 | 2385 |
| openvision.C           | 2386 |
| openvision3.C          | 2388 |
| OptionParser.C         | 2390 |
| OptionParser.H         | 2391 |
| OrientComputeTypes.H   | 2392 |
| <b>pan-tilt.C</b>      | ??   |
| ParamMap.C             | 2394 |
| ParamMap.H             | 2395 |
| PbmParser.C            | 2396 |
| PbmParser.H            | 2397 |
| PfmParser.C            | 2398 |
| PfmParser.H            | 2399 |
| Pixels.H               | 2400 |
| PngParser.C            | 2404 |
| PngParser.H            | 2405 |
| PngWriter.C            | 2406 |
| PngWriter.H            | 2407 |
| Point.H                | 2408 |
| Point2D.H              | 2409 |
| Point2DT.H             | 2412 |
| pointAndFlood.C        | 2413 |
| <b>PointClasses.H</b>  | ??   |
| PolatSagiTest.C        | 2416 |
| prepare_target_image.C | 2421 |
| <b>ProbDist.C</b>      | ??   |
| <b>ProbDist.H</b>      | ??   |
| Promotions.H           | 2422 |
| pSNR.C                 | 2424 |
| psycho-dsearch.C       | 2425 |
| psycho-headmove.C      | 2427 |
| psycho-movie.C         | 2428 |
| psycho-movie2.C        | 2429 |

|                                      |      |
|--------------------------------------|------|
| <b>psycho-noCheat.C</b>              | ??   |
| <a href="#">psycho-obfba.C</a>       | 2431 |
| <b>psycho-obfba2.C</b>               | ??   |
| <b>psycho-only.C</b>                 | ??   |
| <a href="#">psycho-search.C</a>      | 2434 |
| <a href="#">psycho-still.C</a>       | 2436 |
| <a href="#">PsychoDisplay.C</a>      | 2437 |
| <a href="#">PsychoDisplay.H</a>      | 2439 |
| <a href="#">pvision-master.C</a>     | 2441 |
| <a href="#">pvision.C</a>            | 2443 |
| <a href="#">pvisionTCP-defs.H</a>    | 2445 |
| <a href="#">pvisionTCP-master.C</a>  | 2447 |
| <a href="#">pvisionTCP.C</a>         | 2450 |
| <a href="#">pvisionTCP2-master.C</a> | 2454 |
| <a href="#">pvisionTCP2.C</a>        | 2458 |
| <a href="#">pvisionTCP3-master.C</a> | 2462 |
| <a href="#">pvisionTCP3.C</a>        | 2466 |
| <a href="#">pvmdefs.H</a>            | 2470 |
| <a href="#">pvmMessage.C</a>         | 2471 |
| <a href="#">pvmMessage.H</a>         | 2472 |
| <a href="#">Pyramid_Ops.C</a>        | 2474 |
| <a href="#">Pyramid_Ops.H</a>        | 2480 |
| <a href="#">PyramidTypes.H</a>       | 2489 |
| <a href="#">PyrBuilder.C</a>         | 2491 |
| <a href="#">PyrBuilder.H</a>         | 2492 |
| <a href="#">PyrFoveator.C</a>        | 2494 |
| <a href="#">PyrFoveator.H</a>        | 2495 |
| <a href="#">RadioDecoder.C</a>       | 2497 |
| <a href="#">RadioDecoder.H</a>       | 2499 |
| <a href="#">Range.H</a>              | 2500 |
| <a href="#">Raster.C</a>             | 2501 |
| <a href="#">Raster.H</a>             | 2504 |
| <a href="#">RasterFileFormat.H</a>   | 2505 |
| <a href="#">readConfig.C</a>         | 2506 |
| <a href="#">readConfig.H</a>         | 2507 |
| <b>readMatrix.C</b>                  | ??   |
| <b>readMatrix.H</b>                  | ??   |
| <a href="#">Rectangle.H</a>          | 2508 |
| <a href="#">retina.C</a>             | 2509 |
| <a href="#">Retina.C</a>             | 2511 |
| <a href="#">Retina.H</a>             | 2512 |
| <a href="#">SaccadeController.C</a>  | 2513 |
| <a href="#">SaccadeController.H</a>  | 2514 |
| <a href="#">SaccadeControllers.C</a> | 2515 |
| <a href="#">SaccadeControllers.H</a> | 2516 |
| <a href="#">saccades.C</a>           | 2518 |



|                            |      |
|----------------------------|------|
| safecopy.H                 | 2520 |
| sal_count.C                | 2522 |
| saliency.C                 | 2524 |
| saliency.H                 | 2527 |
| SaliencyMap.C              | 2540 |
| SaliencyMap.H              | 2541 |
| Sbqa.C                     | 2543 |
| ScaleSpace.C               | 2544 |
| ScaleSpace.H               | 2546 |
| searcharray.C              | 2548 |
| searcharray2.C             | 2550 |
| searcharray2d.C            | 2552 |
| segmentImage.C             | 2554 |
| segmentImage.H             | 2555 |
| segmentImage2.C            | ??   |
| segmentImage2.H            | ??   |
| segmentImageMC.C           | ??   |
| segmentImageMC.H           | ??   |
| segmentImageMerge.C        | ??   |
| segmentImageMerge.H        | 2556 |
| segmentImageMerge2.C       | ??   |
| segmentImageMerge2.H       | ??   |
| segmentImageTrack.C        | ??   |
| segmentImageTrack.H        | ??   |
| segmentImageTrack2.C       | ??   |
| segmentImageTrack2.H       | ??   |
| segmentImageTrackMC.C      | ??   |
| segmentImageTrackMC.H      | 2558 |
| serial.C                   | 2559 |
| serial.H                   | 2560 |
| ShapeEstimator.C           | 2562 |
| ShapeEstimator.H           | 2564 |
| ShapeEstimatorModes.H      | 2565 |
| shapeEstimatorWebpage.C    | 2568 |
| SharedPtr.H                | 2570 |
| showFilters.C              | 2574 |
| Signature.C                | ??   |
| Signature.H                | ??   |
| SignatureRepository.C      | ??   |
| SignatureRepository.H      | ??   |
| SimulationViewer.C         | 2576 |
| SimulationViewer.H         | 2577 |
| SimulationViewerCompress.C | 2579 |
| SimulationViewerCompress.H | 2582 |
| SimulationViewerEyeMvt.C   | 2583 |
| SimulationViewerEyeMvt.H   | 2585 |

|                           |           |
|---------------------------|-----------|
| SimulationViewerEyeMvt2.C | 2586      |
| SimulationViewerEyeMvt2.H | 2588      |
| SimulationViewerStd.C     | 2589      |
| SimulationViewerStd.H     | 2591      |
| SingleChannel.C           | 2593      |
| SingleChannel.H           | 2595      |
| SingleChannelBeo.C        | 2597      |
| SingleChannelBeo.H        | 2598      |
| SingleChannelBeoServer.C  | 2600      |
| SingleChannelBeoServer.H  | 2601      |
| SingleChannelSurprise.C   | 2602      |
| SingleChannelSurprise.H   | 2603      |
| SockServ.C                | 2604      |
| SockServ.H                | 2606      |
| SoxChannel.C              | 2608      |
| SoxChannel.H              | 2609      |
| SpeedSensor.C             | 2610      |
| SpeedSensor.H             | 2611      |
| <b>src31notes.dxy</b>     | <b>??</b> |
| <b>src3notes.dxy</b>      | <b>??</b> |
| ssc.C                     | 2612      |
| ssc.H                     | 2613      |
| Staircase.C               | 2615      |
| Staircase.H               | 2616      |
| stats.C                   | 2617      |
| stats.H                   | 2618      |
| statsIncl.H               | 2619      |
| StdBrain.C                | 2621      |
| StdBrain.H                | 2623      |
| <b>StereoVision.C</b>     | <b>??</b> |
| stim-noise.C              | 2626      |
| stim-surprise.C           | 2627      |
| streamvision.C            | 2628      |
| streamvision2.C           | 2629      |
| StringConversions.C       | 2631      |
| StringConversions.H       | 2636      |
| SurpriseImage.C           | 2642      |
| SurpriseImage.H           | 2643      |
| SurpriseMap.C             | 2645      |
| SurpriseMap.H             | 2646      |
| SurpriseModel.C           | 2648      |
| SurpriseModel.H           | 2651      |
| TaskRelevanceMap.C        | 2653      |
| TaskRelevanceMap.H        | 2655      |
| TCPcliServ.C              | 2657      |
| TCPcliServ.H              | 2659      |

|                               |           |
|-------------------------------|-----------|
| TCPcommunicator.C             | 2661      |
| TCPcommunicator.H             | 2663      |
| TCPdefs.H                     | 2665      |
| TCPmessage.C                  | 2667      |
| TCPmessage.H                  | 2669      |
| test-align.C                  | 2671      |
| test-Anim.C                   | 2672      |
| test-audioGrab.C              | 2675      |
| test-Beobot.C                 | 2677      |
| test-BeoChip.C                | 2678      |
| test-beoGrab-master.C         | 2679      |
| <b>test-beoGrab.C</b>         | <b>??</b> |
| test-BeoSub.C                 | 2680      |
| test-BeoSubMotor.C            | 2681      |
| <b>test-biasFeatures.C</b>    | <b>??</b> |
| test-binMorph.C               | 2682      |
| test-BitObject.C              | 2683      |
| test-BitObjectExtract.C       | 2684      |
| test-BPnnet.C                 | 2685      |
| <b>test-Cam.C</b>             | <b>??</b> |
| test-cameraMotion.C           | 2690      |
| <b>test-canny.C</b>           | <b>??</b> |
| test-carcontrol.C             | 2692      |
| <b>test-channel.C</b>         | <b>??</b> |
| test-color.C                  | 2694      |
| test-Colors.C                 | 2695      |
| test-colorSegment.C           | 2697      |
| <b>test-colorSegment2.C</b>   | <b>??</b> |
| test-Compare.C                | 2699      |
| test-component.C              | 2700      |
| test-compress.C               | 2701      |
| <b>test-CoordTransforms.C</b> | <b>??</b> |
| test-covEstimate.C            | 2705      |
| test-diskgrab1394.C           | 2707      |
| test-dyncast.C                | 2710      |
| test-EdgeDetect.C             | 2711      |
| <b>test-featureNPC.C</b>      | <b>??</b> |
| test-FFN.C                    | 2712      |
| test-flood.C                  | 2715      |
| test-followColor-slave.C      | 2716      |
| test-followColor.C            | 2718      |
| test-Foveator.C               | 2721      |
| test-gabor.C                  | 2723      |
| test-gabor2.C                 | 2724      |
| test-gabor3.C                 | 2726      |
| test-gabor4.C                 | 2728      |

|                                        |      |
|----------------------------------------|------|
| <b>test-Gist-Sal-Nav.C</b>             | ??   |
| <a href="#">test-gist.C</a>            | 2730 |
| <b>test-gistNav-slave-lcd.C</b>        | ??   |
| <b>test-gistNav-slave.C</b>            | ??   |
| <b>test-gistNav.C</b>                  | ??   |
| <a href="#">test-GPS.C</a>             | 2732 |
| <a href="#">test-grab.C</a>            | 2733 |
| <a href="#">test-gyro.C</a>            | 2735 |
| <a href="#">test-hmax.C</a>            | 2736 |
| <a href="#">test-hmax2.C</a>           | 2739 |
| <a href="#">test-hmax3.C</a>           | 2741 |
| <a href="#">test-hmax4.C</a>           | 2743 |
| <b>test-HMR3300.C</b>                  | ??   |
| <a href="#">test-HSV.C</a>             | 2744 |
| <a href="#">test-hueChannel.C</a>      | 2746 |
| <a href="#">test-Image.C</a>           | 2747 |
| <a href="#">test-ImageDisplay.C</a>    | 2750 |
| <a href="#">test-ImageEqual.C</a>      | 2751 |
| <b>test-IRoom.C</b>                    | ??   |
| <b>test-IRoomAgentMinder.C</b>         | ??   |
| <b>test-IRoomCameraHandoff.C</b>       | ??   |
| <b>test-IRoomCameraHandoff2.C</b>      | ??   |
| <b>test-IRoomCoordTransforms.C</b>     | ??   |
| <b>test-IRoomExampleModule.C</b>       | ??   |
| <b>test-IRoomFoo.C</b>                 | ??   |
| <b>test-IRoomPursuitTracker.C</b>      | ??   |
| <b>test-IRoomSignatureRepository.C</b> | ??   |
| <b>test-IRoomUserInterface.C</b>       | ??   |
| <b>test-itemMask.C</b>                 | ??   |
| <a href="#">test-Jet.C</a>             | 2752 |
| <a href="#">test-junction.C</a>        | 2754 |
| <a href="#">test-KalmanFilter.C</a>    | 2755 |
| <a href="#">test-KeyBoard.C</a>        | 2756 |
| <a href="#">test-keyboardGrab.C</a>    | 2757 |
| <a href="#">test-LevelSpec.C</a>       | 2759 |
| <b>test-lowDec.C</b>                   | ??   |
| <a href="#">test-mbari.C</a>           | 2760 |
| <a href="#">test-monty.C</a>           | 2762 |
| <b>test-moveCamera.C</b>               | ??   |
| <a href="#">test-MPEGStream.C</a>      | 2763 |
| <a href="#">test-multigrab.C</a>       | 2764 |
| <a href="#">test-multiTrack.C</a>      | 2766 |
| <a href="#">test-NPclassify.C</a>      | 2768 |
| <a href="#">test-NPclassify2.C</a>     | 2772 |
| <a href="#">test-optimizeHSV.C</a>     | 2774 |
| <b>test-parseScanpath.C</b>            | ??   |

|                               |      |
|-------------------------------|------|
| test-Pixels.C                 | 2776 |
| <b>test-ProbDist.C</b>        | ??   |
| test-psycho.C                 | 2777 |
| test-radioDecoder.C           | 2778 |
| test-raomodel.C               | 2780 |
| test-Raster.C                 | 2782 |
| test-reset.C                  | 2784 |
| <b>test-roadShape.C</b>       | ??   |
| <b>test-rwBinImage.C</b>      | ??   |
| <b>test-segmentImageMC.C</b>  | ??   |
| <b>test-segmentLandmark.C</b> | ??   |
| test-ShapeEstimator.C         | 2785 |
| test-shift.C                  | 2788 |
| test-smcompress.C             | 2789 |
| test-SoxChannel.C             | 2794 |
| test-SpeedSensor.C            | 2796 |
| test-ssc.C                    | 2797 |
| test-stereo.C                 | 2799 |
| test-surprise.C               | 2800 |
| test-VCC4.C                   | 2801 |
| test-Xwin.C                   | 2803 |
| testimagesat.C                | 2804 |
| TestSuite.C                   | 2805 |
| TestSuite.H                   | 2806 |
| Timer.H                       | 2808 |
| <b>train-gist-scenes.C</b>    | ??   |
| train-gist_headSize.C         | 2811 |
| train-gist_headSize2.C        | 2816 |
| <b>Tree.C</b>                 | ??   |
| <b>Tree.H</b>                 | ??   |
| TypeTraits.H                  | 2819 |
| V4Lgrabber.C                  | 2820 |
| V4Lgrabber.H                  | 2822 |
| V4LgrabberLight.C             | 2825 |
| V4LgrabberLight.H             | 2826 |
| VCC4.C                        | 2827 |
| VCC4.H                        | 2828 |
| VCC4cmddef.H                  | 2830 |
| <b>Vector3.H</b>              | ??   |
| videodiff.C                   | 2834 |
| videograb.C                   | 2835 |
| vision.C                      | 2837 |
| vision.H                      | 2844 |
| VisualBuffer.C                | 2846 |
| VisualBuffer.H                | 2848 |
| VisualCortex.C                | 2849 |

|                            |           |
|----------------------------|-----------|
| VisualCortex.H             | 2851      |
| VisualCortexBeo.C          | 2854      |
| VisualCortexBeo.H          | 2855      |
| VisualCortexConfigurator.C | 2856      |
| VisualCortexConfigurator.H | 2857      |
| VisualCortexEyeMvt.C       | 2858      |
| VisualCortexEyeMvt.H       | 2859      |
| VisualCortexSurprise.C     | 2860      |
| VisualCortexSurprise.H     | 2861      |
| VisualEvent.C              | 2862      |
| VisualEvent.H              | 2864      |
| VisualFeatures.H           | 2866      |
| VisualObject.C             | 2871      |
| VisualObject.H             | 2872      |
| VisualObjectDB.C           | 2874      |
| VisualObjectDB.H           | 2875      |
| webPageHelpers.C           | 2877      |
| webPageHelpers.H           | 2881      |
| WinnerTakeAll.C            | 2885      |
| WinnerTakeAll.H            | 2886      |
| <b>wrap-grab.C</b>         | <b>??</b> |
| WTAwinner.H                | 2888      |
| <b>XML_Parse.C</b>         | <b>??</b> |
| <b>XML_Parse.H</b>         | <b>??</b> |
| XWindow.C                  | 2889      |
| XWindow.H                  | 2890      |
| XWinManaged.C              | 2893      |
| XWinManaged.H              | 2894      |

## Chapter 7

# iLab Neuromorphic Vision Toolkit Page Index

### 7.1 iLab Neuromorphic Vision Toolkit Related Pages

Here is a list of all related documentation pages:

|                                                      |                      |
|------------------------------------------------------|----------------------|
| Building models using the toolkit . . . . .          | <a href="#">2897</a> |
| Changes from Revisions 3.0 to 3.1 in src3/ . . . . . | <a href="#">2902</a> |
| Towards a neural simulation environment . . . . .    | <a href="#">2903</a> |
| Tips and frequently asked questions . . . . .        | <a href="#">2905</a> |
| Changes from src2 to src3 . . . . .                  | <a href="#">2911</a> |
| The make/build process . . . . .                     | <a href="#">2912</a> |
| Ref-counting and copy-on-write . . . . .             | <a href="#">2914</a> |
| Image iterators . . . . .                            | <a href="#">2919</a> |
| Free functions for image operations . . . . .        | <a href="#">2922</a> |
| Test suite(s) . . . . .                              | <a href="#">2924</a> |
| The std:: namespace . . . . .                        | <a href="#">2925</a> |
| Use of std::numeric_limits . . . . .                 | <a href="#">2927</a> |





## Chapter 8

# iLab Neuromorphic Vision Toolkit Module Documentation

### 8.1 max-normalization algorithms

#### Types of normalization (normtyp)

- `#define NBMAXNORMTYPES 10`
- `#define VCXNORM_DEFAULT VCXNORM_FANCY`
- `enum MaxNormType {`  
    `VCXNORM_NONE = 0, VCXNORM_MAXNORM = 1, VCXNORM_-`  
    `FANCY = 2, VCXNORM_FANCYFAST = 3,`  
    `VCXNORM_FANCYONE = 4, VCXNORM_FANCYLANDMARK = 5,`  
    `VCXNORM_LANDMARK = 6, VCXNORM_FANCYWEAK = 7,`  
    `VCXNORM_IGNORE = 8, VCXNORM_SURPRISE = 9 }`
- `const char * maxNormTypeName (const MaxNormType m)`  
    *Get a name in clear for a given type.*

#### Additional parameters for fancynorm versions of the algorithm.

- `const int FANCYITER = 5`  
    *default number of iterations*
- `const double FANCYESIG = 2`  
    *excitatory sigma as of image size*

- const double `FANCYISIG` = 25  
*inhibitory sigma as of image size*
- const double `FANCYCOEX` = 0.5  
*excitatory coefficient (strength)*
- const double `FANCYCOIN` = 1.5  
*inhibitory coefficient (strength)*
- const double `FANCYINHI` = 2.0  
*strength of global inhibition*
- const double `FANCYG` = 2.1  
*for sigmoid normalization*
- const double `FANCYH` = 2.0  
*for sigmoid normalization*
- const double `FANCYS` = 1.0  
*for sigmoid normalization*
- const int `LRLEVEL` = 2  
*for tuned long-range excitation:*

### Standard min/max bounds for `maxNormalize()`.

- const float `MAXNORMMAX` = 10.0f  
*upper bound for `maxNormalize()`*
- const float `MAXNORMMIN` = 0.0f  
*lower bound for `maxNormalize()`*
- const float `MAXNORMLANDMARK` = 255.0f  
*upper bound for `maxNormalizeFancyLandmark()`*

### Functions

- template<class T> `Image< T > maxNormalize` (const `Image< T > &src`, const T mi, const T ma, const `MaxNormType` normtyp=VCXNORM\_DEFAULT, int nbiter=`FANCYITER`, const `Image< float > *lrexcit`=0)

*Generic implementation (select algorithm with normtyp).*

- `template<class T> Image< T > maxNormalizeNone (const Image< T > &src, const T mi, const T ma)`

*Specialized implementation (corresponds to VCXNORM\_NONE).*

- `template<class T> Image< T > maxNormalizeStd (const Image< T > &src, const T mi, const T ma)`

*Specialized implementation (corresponds to VCXNORM\_MAXNORM).*

- `template<class T> Image< T > maxNormalizeFancyFast (const Image< T > &src, const T mi, const T ma, const int nbiter=FANCYITER)`

*Specialized implementation (corresponds to VCXNORM\_FANCYFAST).*

- `template<class T> Image< T > maxNormalizeFancy (const Image< T > &src, const T mi, const T ma, const int nbiter=FANCYITER, const Image< float > *lrexcit=0)`

*Specialized implementation (corresponds to VCXNORM\_FANCY).*

- `template<class T> Image< T > maxNormalizeFancyWeak (const Image< T > &src, const T mi, const T ma)`

*Weak version of FancyOne.*

- `template<class T> Image< T > maxNormalizeFancyLandmark (const Image< T > &src, const T mi, const T ma, const int nbiter=FANCYITER)`

*Specialized implementation (corresponds to VCXNORM\_FANCY).*

- `template<class T> Image< T > maxNormalizeLandmark (const Image< T > &src, const T mi, const T ma)`

*Specialized implementation (corresponds to VCXNORM\_LANDMARK).*

- `template<class T> int findPeaks (const Image< T > &src, const T mi, const T ma, double &sum)`

*to find number of peaks in the image (used in landmark detection)*

- `template<class T> float goodness_map (const Image< T > &src)`

*to find the goodness of a map (used in landmark detection)*

### 8.1.1 Define Documentation

#### 8.1.1.1 `#define NBMAXNORMTYPES 10`

Have this value always equal to the largest of the VCXNORMs (used for range checking)

Definition at line 62 of file fancynorm.H.

Referenced by `convertFromString()`, and `maxNormTypeName()`.

#### 8.1.1.2 `#define VCXNORM_DEFAULT VCXNORM_FANCY`

Always points to the default maxnorm operation

Definition at line 74 of file fancynorm.H.

Referenced by `main()`.

### 8.1.2 Enumeration Type Documentation

#### 8.1.2.1 `enum MaxNormType`

Enumeration values:

`VCXNORM_NONE` no max-normalization, but may change range

`VCXNORM_MAXNORM` non-iterative maxnorm

`VCXNORM_FANCY` full implementation of fancy maxnorm

`VCXNORM_FANCYFAST` fast implementation of fancy maxnorm

`VCXNORM_FANCYONE` one-iteration only of fancy maxnorm

`VCXNORM_FANCYLANDMARK` fancy maxnorm to find landmark

`VCXNORM_LANDMARK` to find landmarks

`VCXNORM_FANCYWEAK` Weak version of FancyOne.

`VCXNORM_IGNORE` Do absolutely nothing, not even change range.

`VCXNORM_SURPRISE` Like ignore but VCX will do something.

Definition at line 46 of file fancynorm.H.

Referenced by `ComplexChannel::ComplexChannel()`, `convertFromString()`, `DirectFeedChannel::DirectFeedChannel()`, `main()`, `SingleChannel::SingleChannel()`, and `VisualCortex::VisualCortex()`.

### 8.1.3 Function Documentation

#### 8.1.3.1 `template<class T> int findPeaks (const Image< T > & src, const T mi, const T ma, double & sum)`

to find number of peaks in the image (used in landmark detection)

Definition at line 505 of file fancynorm.C.

References `Image< T >::findMax()`, `maxNormalize()`, `segmentLandmark()`, and `VCXNORM_FANCY`.

Referenced by `goodness_map()`, and `maxNormalizeLandmark()`.

#### 8.1.3.2 `template<class T> float goodness_map (const Image< T > & src)`

to find the goodness of a map (used in landmark detection)

Definition at line 538 of file fancynorm.C.

References `ASSERT`, `Image< T >::findMax()`, `findPeaks()`, `goodness_map()`, `Image< T >::initialized()`, `MAXNORMLANDMARK`, `MAXNORMMIN`, `Image< T >::normalize()`, and `segmentLandmark()`.

Referenced by `ComplexChannel::combineOutputs()`, `SingleChannel::combineSubMaps()`, `goodness_map()`, and `maxNormalizeLandmark()`.

#### 8.1.3.3 `template<class T> Image<T> maxNormalize (const Image< T > & src, const T mi, const T ma, const MaxNormType normtyp = VCXNORM_DEFAULT, int nbiter = FANCYITER, const Image< float > * lrexcit = 0)`

Generic implementation (select algorithm with *normtyp*).

Normalize between mi and ma and multiply by (ma - mean). Versions with more arguments implement the core within-feature spatial competition for salience. See papers for details.

Definition at line 56 of file fancynorm.C.

References `Image< T >::getMinMax()`, `maxNormalizeFancy()`, `maxNormalizeFancyFast()`, `maxNormalizeFancyLandmark()`, `maxNormalizeFancyWeak()`, `maxNormalizeLandmark()`, `maxNormalizeNone()`, `maxNormalizeStd()`, `VCXNORM_FANCY`, `VCXNORM_FANCYFAST`, `VCXNORM_FANCYLANDMARK`, `VCXNORM_FANCYONE`, `VCXNORM_FANCYWEAK`, `VCXNORM_IGNORE`, `VCXNORM_LANDMARK`, `VCXNORM_MAXNORM`, `VCXNORM_NONE`, and `VCXNORM_SURPRISE`.

Referenced by `DirectFeedChannel::applyMaxNorm()`, `SoxChannel::combine-`

Outputs(), MultiDirectFeedChannel::combineOutputs(), ComplexChannel::combineOutputs(), SingleChannel::combineSubMaps(), HueChannel::combineSubMaps(), computeCMAP(), findPeaks(), VisualCortex::getOutput(), FlickerChannel::getOutput(), SingleChannel::getRawCSmap(), SingleChannel::postProcessMap(), BeobotVisualCortex::process(), and BeobotVisualCortex::processEnd().

#### 8.1.3.4 `template<class T> Image<T> maxNormalizeFancy (const Image< T > & src, const T mi, const T ma, const int nbiter = FANCYITER, const Image< float > * lrexcit = 0)`

Specialized implementation (corresponds to VCXNORM\_FANCY).

fancyNorm from Itti et al, JEI, 2001; FULL implementation.

Definition at line 245 of file fancynorm.C.

References ASSERT, Image< T >::clamp(), convolve(), downSize(), FANCYCOEX, FANCYCOIN, FANCYESIG, FANCYINHI, FANCYISIG, Image< T >::getHeight(), Image< T >::getMinMax(), Image< T >::getWidth(), Image< T >::initialized(), LR-LEVEL, Image< T >::normalize(), Image< T >::rectify(), rescale(), and sqrt().

Referenced by maxNormalize().

#### 8.1.3.5 `template<class T> Image<T> maxNormalizeFancyFast (const Image< T > & src, const T mi, const T ma, const int nbiter = FANCYITER)`

Specialized implementation (corresponds to VCXNORM\_FANCYFAST).

fancyNorm from Itti et al, JEI, 2001; FAST implementation.

Definition at line 187 of file fancynorm.C.

References ASSERT, Image< T >::attenuateBorders(), buildPyrGaussian(), FANCYCOEX, FANCYCOIN, FANCYESIG, FANCYINHI, FANCYISIG, Image< T >::getHeight(), Image< T >::getMinMax(), Image< T >::getWidth(), Image< T >::initialized(), Image< T >::normalize(), Image< T >::rectify(), rescale(), and sqrt().

Referenced by maxNormalize().

#### 8.1.3.6 `template<class T> Image<T> maxNormalizeFancyLandmark (const Image< T > & src, const T mi, const T ma, const int nbiter = FANCYITER)`

Specialized implementation (corresponds to VCXNORM\_FANCY).

fancyNorm from Itti et al, JEI, 2001; FULL implementation. adapted to find landmarks

Definition at line 354 of file fancynorm.C.

References ASSERT, FANCYCOEX, FANCYCOIN, FANCYINHI, Image< T >::getHeight(), Image< T >::getMinMax(), Image< T >::getWidth(), Image< T >::initialized(), Image< T >::normalize(), Image< T >::rectify(), and sqrt().

Referenced by maxNormalize().

#### 8.1.3.7 **template<class T> Image<T> maxNormalizeFancyWeak (const Image< T > & src, const T *mi*, const T *ma*)**

Weak version of FancyOne.

Definition at line 308 of file fancynorm.C.

References ASSERT, FANCYCOEX, FANCYCOIN, FANCYESIG, FANCYINHI, FANCYISIG, Image< T >::getHeight(), Image< T >::getMinMax(), Image< T >::getWidth(), Image< T >::initialized(), Image< T >::normalize(), Image< T >::rectify(), and sqrt().

Referenced by maxNormalize().

#### 8.1.3.8 **template<class T> Image<T> maxNormalizeLandmark (const Image< T > & src, const T *mi*, const T *ma*)**

Specialized implementation (corresponds to VCXNORM\_LANDMARK).

to find landmarks

Definition at line 404 of file fancynorm.C.

References ASSERT, Image< T >::findMax(), findPeaks(), Image< T >::getVal(), goodness\_map(), Image< T >::initialized(), MAXNORMLANDMARK, MAXNORMMIN, Image< T >::normalize(), Image< T >::resize(), and segmentLandmark().

Referenced by maxNormalize().

#### 8.1.3.9 **template<class T> Image<T> maxNormalizeNone (const Image< T > & src, const T *mi*, const T *ma*)**

Specialized implementation (corresponds to VCXNORM\_NONE).

No max-normalization, just normalize between *mi* and *ma*.

Definition at line 113 of file fancynorm.C.

References Image< T >::normalize(), and Image< T >::rectify().

Referenced by maxNormalize().

#### 8.1.3.10 `template<class T> Image<T> maxNormalizeStd (const Image< T > & src, const T mi, const T ma)`

Specialized implementation (corresponds to VCXNORM\_MAXNORM).

maxNorm from Itti et al, IEEE-PAMI, 1998.

Definition at line 129 of file fancynorm.C.

References ASSERT, Image< T >::getHeight(), Image< T >::getVal(), Image< T >::getWidth(), Image< T >::initialized(), Image< T >::normalize(), and Image< T >::rectify().

Referenced by maxNormalize().

#### 8.1.3.11 `const char* maxNormTypeName (const MaxNormType m) [inline]`

Get a name in clear for a given type.

Definition at line 65 of file fancynorm.H.

References NBMAXNORMTYPES.

Referenced by ComplexChannel::combineOutputs(), SingleChannel::combineSubMaps(), HueChannel::combineSubMaps(), convertFromString(), convertToString(), VisualCortex::getOutput(), and SingleChannel::postProcessMap().

### 8.1.4 Variable Documentation

#### 8.1.4.1 `const double FANCYCOEX = 0.5`

excitatory coefficient (strength)

Definition at line 83 of file fancynorm.H.

Referenced by maxNormalizeFancy(), maxNormalizeFancyFast(), maxNormalizeFancyLandmark(), and maxNormalizeFancyWeak().

#### 8.1.4.2 `const double FANCYCOIN = 1.5`

inhibitory coefficient (strength)

Definition at line 84 of file fancynorm.H.

Referenced by maxNormalizeFancy(), maxNormalizeFancyFast(), maxNormalizeFancyLandmark(), and maxNormalizeFancyWeak().



**8.1.4.3 const double FANCYESIG = 2**

excitatory sigma as of image size

Definition at line 81 of file fancynorm.H.

Referenced by maxNormalizeFancy(), maxNormalizeFancyFast(), and maxNormalizeFancyWeak().

**8.1.4.4 const double FANCYG = 2.1**

for sigmoid normalization

Definition at line 87 of file fancynorm.H.

**8.1.4.5 const double FANCYH = 2.0**

for sigmoid normalization

Definition at line 88 of file fancynorm.H.

**8.1.4.6 const double FANCYINHI = 2.0**

strength of global inhibition

Definition at line 85 of file fancynorm.H.

Referenced by maxNormalizeFancy(), maxNormalizeFancyFast(), maxNormalizeFancyLandmark(), and maxNormalizeFancyWeak().

**8.1.4.7 const double FANCYISIG = 25**

inhibitory sigma as of image size

Definition at line 82 of file fancynorm.H.

Referenced by maxNormalizeFancy(), maxNormalizeFancyFast(), and maxNormalizeFancyWeak().

**8.1.4.8 const int FANCYITER = 5**

default number of iterations

Definition at line 80 of file fancynorm.H.

**8.1.4.9 const double [FANCYS](#) = 1.0**

for sigmoid normalization

Definition at line 89 of file fancynorm.H.

**8.1.4.10 const int [LRLEVEL](#) = 2**

for tuned long-range excitation:

Definition at line 91 of file fancynorm.H.

Referenced by `maxNormalizeFancy()`.

**8.1.4.11 const float [MAXNORMLANDMARK](#) = 255.0f**

upper bound for `maxNormalizeFancyLandmark()`

Definition at line 103 of file fancynorm.H.

Referenced by `goodness_map()`, and `maxNormalizeLandmark()`.

**8.1.4.12 const float [MAXNORMMAX](#) = 10.0f**

upper bound for `maxNormalize()`

Definition at line 99 of file fancynorm.H.

Referenced by `ChannelBase::ChannelBase()`, `SoxChannel::combineOutputs()`, `computeCMAP()`, and `SingleChannel::postProcessMap()`.

**8.1.4.13 const float [MAXNORMMIN](#) = 0.0f**

lower bound for `maxNormalize()`

Definition at line 101 of file fancynorm.H.

Referenced by `ChannelBase::ChannelBase()`, `SoxChannel::combineOutputs()`, `computeCMAP()`, `goodness_map()`, `maxNormalizeLandmark()`, and `SingleChannel::postProcessMap()`.

## 8.2 String conversion helpers

### Generic string conversion functions

- template<class T> void `convertToString` (const T &val, `std::string` &str)
- template<class T> bool `convertFromString` (const `std::string` &str, T &val)

### Specialized implementations of string conversion functions

- void `convertToString` (const `std::string` &val, `std::string` &str)  
*`std::string` specialization*
- bool `convertFromString` (const `std::string` &str, `std::string` &val)  
*`std::string` specialization*
- void `convertToString` (const `Dims` &val, `std::string` &str)  
*`Dims` specialization: format is "<int>x<int>"*
- bool `convertFromString` (const `std::string` &str, `Dims` &val)  
*`Dims` specialization: format is "<int>x<int>"*
- void `convertToString` (const `Point2D` &val, `std::string` &str)  
*`Point2D` specialization: format is "<int>,<int>"*
- bool `convertFromString` (const `std::string` &str, `Point2D` &val)  
*`Point2D` specialization: format is "<int>,<int>"*
- void `convertToString` (const bool &val, `std::string` &str)  
*bool specialization: format is "true|false"*
- bool `convertFromString` (const `std::string` &str, bool &val)  
*bool specialization*
- void `convertToString` (const `FrameGrabberMode` val, `std::string` &str)  
*Conversion from `FrameGrabberMode` to string.*
- bool `convertFromString` (const `std::string` &str, `FrameGrabberMode` &val)  
*Conversion from string to `FrameGrabberMode`.*
- void `convertToString` (const `FrameRange` &val, `std::string` &str)  
*`FrameRange` specialization: format is "<int>-<int>@<float>"*

- bool `convertFromString` (const `std::string` &str, `FrameRange` &val)  
*FrameRange* specialization: format is "<int>-<int>@<float>".
- void `convertToString` (const `LevelSpec` &val, `std::string` &str)  
*LevelSpec* specialization: format is "<int>-<int>,<int>-<int>,<int>".
- bool `convertFromString` (const `std::string` &str, `LevelSpec` &val)  
*LevelSpec* specialization: format is "<int>-<int>,<int>-<int>,<int>".
- void `convertToString` (const `Rectangle` &val, `std::string` &str)  
*Rectangle* specialization: format is "<int>,<int>,<int>,<int>".
- bool `convertFromString` (const `std::string` &str, `Rectangle` &val)  
*Rectangle* specialization: format is "<int>,<int>,<int>,<int>".
- void `convertToString` (const `RasterFileFormat` val, `std::string` &str)  
*RasterFileFormat* specialization.
- bool `convertFromString` (const `std::string` &str, `RasterFileFormat` &val)  
*RasterFileFormat* specialization.
- void `convertToString` (const `PixRGB< byte >` &val, `std::string` &str)  
*PixRGB<byte>* specialization.
- bool `convertFromString` (const `std::string` &str, `PixRGB< byte >` &val)  
*RasterFileFormat* specialization.
- void `convertToString` (const `VisualFeature` val, `std::string` &str)  
*VisualFeature* specialization.
- bool `convertFromString` (const `std::string` &str, `VisualFeature` &val)  
*VisualFeature* specialization.
- void `convertToString` (const `PyramidType` val, `std::string` &str)  
*PyramidType* specialization.
- bool `convertFromString` (const `std::string` &str, `PyramidType` &val)  
*PyramidType* specialization.
- void `convertToString` (const `MaxNormType` val, `std::string` &str)  
*MaxNormType* specialization.

- bool `convertFromString` (const `std::string` &str, `MaxNormType` &val)  
*MaxNormType specialization.*
- void `convertToString` (const `IORtype` val, `std::string` &str)  
*IORtype specialization.*
- bool `convertFromString` (const `std::string` &str, `IORtype` &val)  
*IORtype specialization.*
- void `convertToString` (const `ShapeEstimatorMode` val, `std::string` &str)  
*ShapeEstimatorMode specialization.*
- bool `convertFromString` (const `std::string` &str, `ShapeEstimatorMode` &val)  
*ShapeEstimatorMode specialization.*
- void `convertToString` (const `ShapeEstimatorSmoothMethod` val, `std::string` &str)  
*ShapeEstimatorSmoothType specialization.*
- bool `convertFromString` (const `std::string` &str, `ShapeEstimatorSmoothMethod` &val)  
*ShapeEstimatorSmoothType specialization.*
- void `convertToString` (const `OrientComputeType` val, `std::string` &str)  
*OrientComputeType specialization.*
- bool `convertFromString` (const `std::string` &str, `OrientComputeType` &val)  
*OrientComputeType specialization.*
- void `convertToString` (const `in_addr` &val, `std::string` &str)  
*in\_addr specialization: format is "<int>.<int>.<int>.<int>"*
- bool `convertFromString` (const `std::string` &str, `in_addr` &val)  
*in\_addr specialization: format is "<int>.<int>.<int>.<int>"*
- void `convertToString` (const `BitObjectDrawMode` val, `std::string` &str)  
*BitObjectDrawMode specialization.*
- bool `convertFromString` (const `std::string` &str, `BitObjectDrawMode` &val)  
*BitObjectDrawMode specialization.*
- void `convertToString` (const `ImageDisplayType` val, `std::string` &str)  
*ImageDisplayType specialization.*

- bool `convertFromString` (const `std::string` &str, `ImageDisplayType` &val)  
*ImageDisplayType specialization.*
- void `convertToString` (const `ColorComputeType` val, `std::string` &str)  
*ColorComputeType specialization.*
- bool `convertFromString` (const `std::string` &str, `ColorComputeType` &val)  
*ColorComputeType specialization.*

## Easier-syntax functions to convert to/from string

- template<class T> `std::string toStr` (const T &val)  
*converts val into a string and returns the string*
- template<class T> T `fromStr` (const `std::string` &str)  
*converts str into a T value and returns the T value*

### 8.2.1 Detailed Description

We define a number of string conversion helpers here just so that we don't require each type that we may want to instantiate into a `ModelParam` derivative to define operator<< and operator>>. The main reason is that this definition requires #including (especially if the class is all-inline like `Point2D`), which would largely increase the compile time if we did it for all our simple types like `Dims`, `Point2D`, etc. But the code below may some day migrate into proper definitions of operator<< and operator>> for those data types.

### 8.2.2 Function Documentation

#### 8.2.2.1 bool `convertFromString` (const `std::string` & str, `ColorComputeType` & val)

`ColorComputeType` specialization.

Format is "name" as defined by `colorComputeTypeName()` in `ColorComputeTypes.H`

Definition at line 422 of file `StringConversions.C`.

References `ColorComputeType`, and `NBCOLORCOMPUTETYPES`.

Referenced by `fromStr()`, `main()`, `ModelManager::printHelp()`, and `ModelParam< T >::setValString()`.

### 8.2.2.2 `bool convertFromString (const std::string & str, ImageDisplayType & val)`

[ImageDisplayType](#) specialization.

Format is "name" as defined by `imageDisplayTypeName()` in [ImageDisplayTypes.H](#)

Definition at line 406 of file `StringConversions.C`.

References [ImageDisplayType](#), and `NBIMAGEDISPLAYTYPES`.

### 8.2.2.3 `bool convertFromString (const std::string & str, BitObjectDrawMode & val)`

[BitObjectDrawMode](#) specialization.

Format is "name" as defined in [BitObjectDrawModes.H](#)

Definition at line 390 of file `StringConversions.C`.

References [BitObjectDrawMode](#), `bitObjectDrawModeName()`, and `NBBITOBJECT-DRAWMODES`.

### 8.2.2.4 `bool convertFromString (const std::string & str, in_addr & val)`

`in_addr` specialization: format is "<int>.<int>.<int>.<int>"

Definition at line 375 of file `StringConversions.C`.

### 8.2.2.5 `bool convertFromString (const std::string & str, OrientComputeType & val)`

[OrientComputeType](#) specialization.

Format is "name" as defined in [OrientComputeTypes.H](#)

Definition at line 355 of file `StringConversions.C`.

References `NBORIENTCOMPUTETYPES`, and [OrientComputeType](#).

### 8.2.2.6 `bool convertFromString (const std::string & str, ShapeEstimatorSmoothMethod & val)`

[ShapeEstimatorSmoothType](#) specialization.

Format is "name" as defined in [ShapeEstimatorModes.H](#)

Definition at line 337 of file `StringConversions.C`.

References NBSHAPEESTIMATORSMOOTHMETHODS, ShapeEstimatorSmoothMethod, and shapeEstimatorSmoothMethodName().

#### 8.2.2.7 **bool convertFromString (const [std::string](#) & str, [ShapeEstimatorMode](#) & val)**

ShapeEstimatorMode specialization.

Format is "name" as defined in [ShapeEstimatorModes.H](#)

Definition at line 320 of file StringConversions.C.

References NBSHAPEESTIMATORMODES, ShapeEstimatorMode, and shapeEstimatorModeName().

#### 8.2.2.8 **bool convertFromString (const [std::string](#) & str, [IORtype](#) & val)**

IORtype specialization.

Format is "name" as defined by iorTypeName() in [IORtypes.H](#)

Definition at line 304 of file StringConversions.C.

References IORtype, and NBIORTYPES.

#### 8.2.2.9 **bool convertFromString (const [std::string](#) & str, [MaxNormType](#) & val)**

MaxNormType specialization.

Format is "name" as defined by [maxNormTypeName\(\)](#) in MaxNormTypes.H

Definition at line 288 of file StringConversions.C.

References MaxNormType, maxNormTypeName(), and NBMAXNORMTYPES.

#### 8.2.2.10 **bool convertFromString (const [std::string](#) & str, [PyramidType](#) & val)**

PyramidType specialization.

Format is "name" as defined by pyramidTypeName() in [PyramidTypes.H](#)

Definition at line 272 of file StringConversions.C.

References NBPYRAMIDTYPES, and PyramidType.

#### 8.2.2.11 **bool convertFromString (const [std::string](#) & str, [VisualFeature](#) & val)**

VisualFeature specialization.



Format is "name" as defined by [featureName\(\)](#) in [VisualFeatures.H](#)

Definition at line 256 of file StringConversions.C.

References [featureName\(\)](#), [NBVISUALFEATURES](#), and [VisualFeature](#).

#### 8.2.2.12 **bool convertFromString** (const [std::string](#) & *str*, [PixRGB](#)< [byte](#) > & *val*)

[RasterFileFormat](#) specialization.

Format is "<byte>,<byte>,<byte>" for R, G, B, or "#RRGGBB" with the RR, GG and BB values in hexadecimal.

Definition at line 231 of file StringConversions.C.

References [PixRGB](#)< [T](#) >::set(), [PixRGB](#)< [T](#) >::setBlue(), [PixRGB](#)< [T](#) >::setGreen(), and [PixRGB](#)< [T](#) >::setRed().

#### 8.2.2.13 **bool convertFromString** (const [std::string](#) & *str*, [RasterFileFormat](#) & *val*)

[RasterFileFormat](#) specialization.

The names used in the definition of the enum [RasterFileFormat](#) will be returned

Definition at line 210 of file StringConversions.C.

#### 8.2.2.14 **bool convertFromString** (const [std::string](#) & *str*, [Rectangle](#) & *val*)

[Rectangle](#) specialization: format is "<int>,<int>,<int>,<int>".

Definition at line 190 of file StringConversions.C.

References [Rectangle](#)::modify().

#### 8.2.2.15 **bool convertFromString** (const [std::string](#) & *str*, [LevelSpec](#) & *val*)

[LevelSpec](#) specialization: format is "<int>-<int>,<int>-<int>,<int>".

Definition at line 171 of file StringConversions.C.

References [LevelSpec](#)::init().

#### 8.2.2.16 **bool convertFromString** (const [std::string](#) & *str*, [FrameRange](#) & *val*)

[FrameRange](#) specialization: format is "<int>-<int>@<float>".

Definition at line 151 of file StringConversions.C.

References `FrameRange::delay`, `FrameRange::first`, and `FrameRange::last`.

#### **8.2.2.17 `bool convertFromString (const std::string & str, FrameGrabberMode & val)`**

Conversion from string to `FrameGrabberMode`.

The names used in the definition of the enum `FrameGrabberMode` (minus the "GM" prefix) will be returned

Definition at line 135 of file StringConversions.C.

References `FrameGrabberMode`, `frameGrabberModeName()`, and `NBFRAMEGRABBERMODES`.

#### **8.2.2.18 `bool convertFromString (const std::string & str, bool & val)`**

`bool` specialization

Anything with a first letter 't', 'T', 'y', 'Y', or '1' will convert to true; anything with a first letter 'f', 'F', 'n', 'N', or '0' will convert to false

Definition at line 119 of file StringConversions.C.

#### **8.2.2.19 `bool convertFromString (const std::string & str, Point2D & val)`**

[Point2D](#) specialization: format is "<int>,<int>".

Definition at line 105 of file StringConversions.C.

References `Point2D::i`, and `Point2D::j`.

#### **8.2.2.20 `bool convertFromString (const std::string & str, Dims & val)`**

[Dims](#) specialization: format is "<int>x<int>".

Definition at line 91 of file StringConversions.C.

#### **8.2.2.21 `bool convertFromString (const std::string & str, std::string & val)`**

[std::string](#) specialization

Definition at line 83 of file StringConversions.C.

**8.2.2.22** `template<class T> bool convertFromString (const std::string & str, T & val) [inline]`

inefficient function to convert from an [std::string](#) into anything that supports operator>>

Definition at line 75 of file StringConversions.C.

**8.2.2.23** `void convertToString (const ColorComputeType val, std::string & str)`

[ColorComputeType](#) specialization.

Format is "name" as defined by [colorComputeTypeName\(\)](#) in [ColorComputeTypes.H](#)

Definition at line 418 of file StringConversions.C.

Referenced by `ModelParam< T >::getValString()`, and `toStr()`.

**8.2.2.24** `void convertToString (const ImageDisplayType val, std::string & str)`

[ImageDisplayType](#) specialization.

Format is "name" as defined by [imageDisplayTypeName\(\)](#) in [ImageDisplayTypes.H](#)

Definition at line 402 of file StringConversions.C.

**8.2.2.25** `void convertToString (const BitObjectDrawMode val, std::string & str)`

[BitObjectDrawMode](#) specialization.

Format is "name" as defined in [BitObjectDrawModes.H](#)

Definition at line 386 of file StringConversions.C.

References [bitObjectDrawModeName\(\)](#).

**8.2.2.26** `void convertToString (const in\_addr & val, std::string & str)`

[in\\_addr](#) specialization: format is "<int>.<int>.<int>.<int>"

Definition at line 367 of file StringConversions.C.

**8.2.2.27** `void convertToString (const OrientComputeType val, std::string & str)`

[OrientComputeType](#) specialization.

Format is "name" as defined in [OrientComputeTypes.H](#)

Definition at line 351 of file StringConversions.C.

**8.2.2.28 void convertToString (const [ShapeEstimatorSmoothMethod](#) val, [std::string](#) & str)**

ShapeEstimatorSmoothType specialization.

Format is "name" as defined in [ShapeEstimatorModes.H](#)

Definition at line 332 of file StringConversions.C.

References shapeEstimatorSmoothMethodName().

**8.2.2.29 void convertToString (const [ShapeEstimatorMode](#) val, [std::string](#) & str)**

ShapeEstimatorMode specialization.

Format is "name" as defined in [ShapeEstimatorModes.H](#)

Definition at line 316 of file StringConversions.C.

References shapeEstimatorModeName().

**8.2.2.30 void convertToString (const [IORtype](#) val, [std::string](#) & str)**

IORtype specialization.

Format is "name" as defined by iorTypeName() in [IORtypes.H](#)

Definition at line 300 of file StringConversions.C.

**8.2.2.31 void convertToString (const [MaxNormType](#) val, [std::string](#) & str)**

MaxNormType specialization.

Format is "name" as defined by maxNormTypeName() in MaxNormTypes.H

Definition at line 284 of file StringConversions.C.

References maxNormTypeName().

**8.2.2.32 void convertToString (const [PyramidType](#) val, [std::string](#) & str)**

PyramidType specialization.

Format is "name" as defined by pyramidTypeName() in [PyramidTypes.H](#)

Definition at line 268 of file StringConversions.C.

**8.2.2.33 void convertToString (const [VisualFeature](#) val, std::string & str)**

[VisualFeature](#) specialization.

Format is "name" as defined by [featureName\(\)](#) in [VisualFeatures.H](#)

Definition at line 252 of file [StringConversions.C](#).

References [featureName\(\)](#).

**8.2.2.34 void convertToString (const [PixRGB](#)< byte > & val, std::string & str)**

[PixRGB](#)<byte> specialization.

Format is "<byte>,<byte>,<byte>" for R, G, B

Definition at line 223 of file [StringConversions.C](#).

References [PixRGB](#)< T >::blue(), [PixRGB](#)< T >::green(), and [PixRGB](#)< T >::red().

**8.2.2.35 void convertToString (const [RasterFileFormat](#) val, std::string & str)**

[RasterFileFormat](#) specialization.

Use it, for example, if you got a file format as text from the command line and need to convert that to a [RasterFileFormat](#) value for a [ModelParam](#). Text strings should match the symbolic names of the [RasterFileFormat](#) enum.

Definition at line 201 of file [StringConversions.C](#).

**8.2.2.36 void convertToString (const [Rectangle](#) & val, std::string & str)**

[Rectangle](#) specialization: format is "<int>,<int>,<int>,<int>".

Format corresponds to "xopleft,yopleft,xbottomright,ybottomright"

Definition at line 182 of file [StringConversions.C](#).

References [Rectangle](#)::bottom(), [Rectangle](#)::left(), [Rectangle](#)::right(), and [Rectangle](#)::top().

**8.2.2.37 void convertToString (const [LevelSpec](#) & val, std::string & str)**

[LevelSpec](#) specialization: format is "<int>-<int>,<int>-<int>,<int>".

Format corresponds to "levmin-levmax,delmin-delmax,maplevel"

Definition at line 162 of file [StringConversions.C](#).

References LevelSpec::delMax(), LevelSpec::delMin(), LevelSpec::levMax(), LevelSpec::levMin(), and LevelSpec::mapLevel().

#### 8.2.2.38 void convertToString (const FrameRange & val, std::string & str)

FrameRange specialization: format is "<int>-<int>@<float>".

Format corresponds to "first-last@delay"

Definition at line 147 of file StringConversions.C.

References FrameRange::delay, FrameRange::first, and FrameRange::last.

#### 8.2.2.39 void convertToString (const FrameGrabberMode val, std::string & str)

Conversion from FrameGrabberMode to string.

Use it, for example, if you got a grabmode as text from the command line and need to convert that to a FrameGrabberMode value for the constructor. Text strings should match the symbolic names of the FrameGrabberMode enum, minus the "GM" prefix.

Definition at line 131 of file StringConversions.C.

References frameGrabberModeName().

#### 8.2.2.40 void convertToString (const bool & val, std::string & str)

bool specialization: format is "true|false"

Definition at line 115 of file StringConversions.C.

#### 8.2.2.41 void convertToString (const Point2D & val, std::string & str)

Point2D specialization: format is "<int>,<int>".

Definition at line 101 of file StringConversions.C.

References Point2D::i, and Point2D::j.

#### 8.2.2.42 void convertToString (const Dims & val, std::string & str)

Dims specialization: format is "<int>x<int>".

Definition at line 87 of file StringConversions.C.

References Dims::h(), and Dims::w().

**8.2.2.43 void convertToString (const [std::string](#) & val, [std::string](#) & str)**

[std::string](#) specialization

Definition at line 79 of file StringConversions.C.

**8.2.2.44 template<class T> void convertToString (const T & val, [std::string](#) & str)**

inefficient function to convert anything that supports operator<< into an [std::string](#)

Definition at line 70 of file StringConversions.C.

**8.2.2.45 template<class T> T fromStr (const [std::string](#) & str)**

converts str into a T value and returns the T value

Definition at line 444 of file StringConversions.C.

References convertFromString().

**8.2.2.46 template<class T> [std::string](#) toStr (const T & val)**

converts val into a string and returns the string

Definition at line 435 of file StringConversions.C.

References convertToString().

Referenced by DirectFeedChannel::DirectFeedChannel(), DirectFeedChannel::doInput(), VisualEventSet::drawTokens(), and main().





## Chapter 9

# iLab Neuromorphic Vision Toolkit Namespace Documentation

### 9.1 Png Namespace Reference

Namespace for functions related to PNG-format image files.

#### Functions

- void `writeGray` (const char \*`filename`, const `Image`< `byte` > &img)

*Write an 8-bit grayscale PNG image file.*

- void `writeRGB` (const char \*`filename`, const `Image`< `PixRGB`< `byte` > > &img)

*Write an 8-bit color PNG image file.*

#### 9.1.1 Detailed Description

Namespace for functions related to PNG-format image files.

## 9.1.2 Function Documentation

### 9.1.2.1 `void Png::writeGray (const char * filename, const Image< byte > & img)`

Write an 8-bit grayscale PNG image file.

Definition at line 210 of file PngWriter.C.

References PngWriter::open(), and PngWriter::writeGray().

### 9.1.2.2 `void Png::writeRGB (const char * filename, const Image< PixRGB< byte > > & img)`

Write an 8-bit color PNG image file.

Definition at line 217 of file PngWriter.C.

References PngWriter::open(), and PngWriter::writeRGB().

## 9.2 Raster Namespace Reference

For reading, writing and displaying images.

### Classes

- class [MetaData](#)  
*Metadata information (typically obtained from GPS).*

### Enumerations

- enum [ColorMode](#) { **MODE\_GREYSCALE**, **MODE\_RGB** }  
*Raster color modes.*
- enum [NormalizeMode](#) { **NO\_NORM** = 0, **STD\_NORM** = 1, **NORM\_WITH\_SCALE** = 2 }  
*Normalization modes when saving float images.*

### Functions

- void [WriteGray](#) (const [Image](#)< byte > &image, const [RasterFileFormat](#) ftype, const char \*fname,...)  
*Write a grayscale byte image as greyscale - pgm format.*
- void [WriteFloat](#) (const [Image](#)< float > &image, const [RasterFileFormat](#) ftype, const bool normalize, const char \*fname,...)  
*Write a float image, normalize if necessary - pgm format.*
- void [WriteFloatNorm](#) (const [Image](#)< float > &image, const [RasterFileFormat](#) ftype, [NormalizeMode](#) nmode, const char \*fname,...)  
*Write a float image with optional normalization style.*
- void [WriteRGB](#) (const [Image](#)< [PixRGB](#)< byte > > &image, const [RasterFileFormat](#) ftype, const char \*fname,...)  
*Write an RGB byte image - ppm format.*
- [Dims](#) [getImageDims](#) (const [RasterFileFormat](#) ftype, const [Raster::ColorMode](#) cmode, const char \*fname,...)  
*Get image dimensions without fully reading the image.*

- [MetaData](#) [getImageMeta](#) (const [RasterFileFormat](#) ftype, const [Raster::Color-Mode](#) cmode, const char \*fname,...)  
*Get image Meta Data without fully reading the image (PPM only).*
- [Image](#)< [byte](#) > [ReadGray](#) (const [RasterFileFormat](#) ftype, const char \*fname,...)  
*Read a greyscale byte image from a greyscale pgm file.*
- [Image](#)< [float](#) > [ReadFloat](#) (const [RasterFileFormat](#) ftype, const char \*fname,...)  
*Read a greyscale float image from a greyscale file.*
- [Image](#)< [PixRGB](#)< [byte](#) > > [ReadRGB](#) (const [RasterFileFormat](#) ftype, const char \*fname,...)  
*Read an RGB byte image from a ppm file.*
- void [NoXV](#) ()  
*Turn off the use of xv.*
- void [Display](#) (const char \*file)  
*calls xv to display an image that is on disk*
- void [Display](#) (const char \*file, const [RasterFileFormat](#) ft, const [Raster::Color-Mode](#) cmode)  
*calls xv to display an image that is on disk, guesses file extension*
- void [waitForKey](#) ()  
*wait for the [return] key*
- template<class T> void [Visu](#) (const [Image](#)< T > &image, const [RasterFileFormat](#) ft, const bool normalize, const bool wdel, const char \*fname,...)  
*Save image of type Image<T> to disk then display it using xv.*
- template<class T> void [Visu](#) (const [Image](#)< [PixRGB](#)< T > > &image, const [RasterFileFormat](#) ft, const bool normalize, const bool wdel, const char \*fname,...)  
*Save image of type Image< PixRGB<T> > to disk then display it using xv.*
- template<class T> void [VisuArray](#) (const [Image](#)< T > \*image, const int nbima, const int Nx, const [RasterFileFormat](#) ftype, const bool normalize, const bool waitdel, const char \*fname,...)  
*Save an array of images of type Image<T>.*

- `template<class T> void VisuArray (const Image< PixRGB< T > > *image, const int nbima, const int Nx, const RasterFileFormat ftype, const bool normalize, const bool waitdel, const char *fname,...)`

*Save an array of images of type Image< PixRGB<T> >.*

### 9.2.1 Detailed Description

For reading, writing and displaying images.

Raster works with PGM files (for grayscale images) and PPM files (for RGB images). Selection between PGM and PPM given the PNM RasterFileFormat is automatic based on whether the image is greyscale or color. In addition, it offers PNG read capability (grayscale, RGB, or color-palette, at any bit-depth), and write capability (8-bit grayscale and RGB). Similarly, it offers write capability for RAWIMAGE data (straight pixel data with no header information), but no such read capability. Both the raw and ASCII variants of PGM/PPM files are supported. Display of images uses the program "xv" to output images to the screen. The filename argument is passed to the various methods of this class in the same way as in printf().

### 9.2.2 Enumeration Type Documentation

#### 9.2.2.1 enum Raster::ColorMode

Raster color modes.

Definition at line 68 of file Raster.H.

#### 9.2.2.2 enum Raster::NormalizeMode

Normalization modes when saving float images.

**Enumeration values:**

**NO\_NORM** no normalization

**STD\_NORM** norm to [0,255]

**NORM\_WITH\_SCALE** norm to [0,255]; text shows original min+max

Definition at line 71 of file Raster.H.

### 9.2.3 Function Documentation

#### 9.2.3.1 **void Raster::Display** (const char \* *file*, const [RasterFileFormat](#) *ft*, const [Raster::ColorMode](#) *cmode*)

calls xv to display an image that is on disk, guesses file extension

Definition at line 596 of file Raster.C.

References [Display\(\)](#).

#### 9.2.3.2 **void Raster::Display** (const char \* *file*)

calls xv to display an image that is on disk

Definition at line 585 of file Raster.C.

Referenced by [Display\(\)](#), [InputFrameSeries::readGray\(\)](#), [InputFrameSeries::readRGB\(\)](#), [OutputFrameSeries::writeFloat\(\)](#), [OutputFrameSeries::writeGray\(\)](#), and [OutputFrameSeries::writeRGB\(\)](#).

#### 9.2.3.3 **Dims Raster::getImageDims** (const [RasterFileFormat](#) *ftype*, const [Raster::ColorMode](#) *cmode*, const char \* *fname*, ...)

Get image dimensions without fully reading the image.

Definition at line 474 of file Raster.C.

References [PngParser::getDims\(\)](#), [PbmParser::getDims\(\)](#), and [RasterFileFormat](#).

Referenced by [InputFrameSeries::getImageDims\(\)](#).

#### 9.2.3.4 **[Raster::MetaData](#) Raster::getImageMeta** (const [RasterFileFormat](#) *ftype*, const [Raster::ColorMode](#) *cmode*, const char \* *fname*, ...)

Get image Meta Data without fully reading the image (PPM only).

Definition at line 502 of file Raster.C.

References [PbmParser::getComments\(\)](#), and [RasterFileFormat](#).

#### 9.2.3.5 **void Raster::NoXV** ()

Turn off the use of xv.

Turn off xv, so that calls to [Display\(\)](#) do nothing, and calls to [Visu\(\)](#) just save the image to disk without displaying it onscreen.

Definition at line 579 of file Raster.C.

Referenced by main().

#### 9.2.3.6 **Image< float > Raster::ReadFloat** (const **RasterFileFormat** *f*type, const char \* *f*name, ...)

Read a greyscale float image from a greyscale file.

Definition at line 546 of file Raster.C.

#### 9.2.3.7 **Image< byte > Raster::ReadGray** (const **RasterFileFormat** *f*type, const char \* *f*name, ...)

Read a greyscale byte image from a greyscale pgm file.

Definition at line 536 of file Raster.C.

Referenced by TaskRelevanceMapStd::input(), and main().

#### 9.2.3.8 **Image< PixRGB< byte > > Raster::ReadRGB** (const **RasterFileFormat** *f*type, const char \* *f*name, ...)

Read an RGB byte image from a ppm file.

Definition at line 564 of file Raster.C.

Referenced by main(), RetinaStd::start2(), and VisualObject::VisualObject().

#### 9.2.3.9 **template<class T> void Raster::Visu** (const **Image< PixRGB< T > >** & *image*, const **RasterFileFormat** *ft*, const bool *normalize*, const bool *w*del, const char \* *f*name, ...)

Save image of type **Image< PixRGB<T> >** to disk then display it using xv.

Definition at line 684 of file Raster.C.

References image.

#### 9.2.3.10 **template<class T> void Raster::Visu** (const **Image< T >** & *image*, const **RasterFileFormat** *ft*, const bool *normalize*, const bool *w*del, const char \* *f*name, ...)

Save image of type **Image<T>** to disk then display it using xv.

Definition at line 673 of file Raster.C.

References image.

Referenced by main(), Omni< T >::run(), and CINNIC::viewNeuronTemplate().

**9.2.3.11** `template<class T> void Raster::VisuArray (const Image< PixRGB< T > > * image, const int nbima, const int Nx, const RasterFileFormat ftype, const bool normalize, const bool waitdel, const char * fname, ...)`

Save an array of images of type Image< PixRGB<T> >.

save array of images and display them together with same normalization put Nx images in every line of the array, and as many lines as necessary

Definition at line 709 of file Raster.C.

References concatArray(), and image.

**9.2.3.12** `template<class T> void Raster::VisuArray (const Image< T > * image, const int nbima, const int Nx, const RasterFileFormat ftype, const bool normalize, const bool waitdel, const char * fname, ...)`

Save an array of images of type Image<T>.

Save array of images and display them together with same normalization put Nx images in every line of the array, and as many lines as necessary

Definition at line 695 of file Raster.C.

References concatArray(), and image.

**9.2.3.13** `void Raster::waitForKey ()`

wait for the [return] key

Definition at line 605 of file Raster.C.

**9.2.3.14** `void Raster::WriteFloat (const Image< float > & image, const RasterFileFormat ftype, const bool normalize, const char * fname, ...)`

Write a float image, normalize if necessary - pgm format.

Definition at line 422 of file Raster.C.

References image, and Image< T >::normalize().

Referenced by main(), and OutputFrameSeries::writeFloat().



**9.2.3.15** `void Raster::WriteFloatNorm (const Image< float > & image, const RasterFileFormat ftype, NormalizeMode nmode, const char * fname, ...)`

Write a float image with optional normalization style.

Definition at line 445 of file Raster.C.

References image, NO\_NORM, NORM\_WITH\_SCALE, Image< T >::normalize(), normalizeWithScale(), and STD\_NORM.

**9.2.3.16** `void Raster::WriteGray (const Image< byte > & image, const RasterFileFormat ftype, const char * fname, ...)`

Write a grayscale byte image as greyscale - pgm format.

Definition at line 415 of file Raster.C.

References image.

Referenced by ImageDisplay::display(), main(), and OutputFrameSeries::writeGray().

**9.2.3.17** `void Raster::WriteRGB (const Image< PixRGB< byte > > & image, const RasterFileFormat ftype, const char * fname, ...)`

Write an RGB byte image - ppm format.

Definition at line 556 of file Raster.C.

References image.

Referenced by BeoSub::checkpoint(), ImageDisplay::display(), contourRun::dump-EnergySigmoid(), main(), VisualObject::save(), VisualObjectMatch::writeOutput(), and OutputFrameSeries::writeRGB().

## 9.3 std Namespace Reference

The Standard Template Library (STL).

### Classes

- class [vector](#)  
*The STL vector container class. Use with `#include` .*
- class [deque](#)  
*The STL deque container class. Use with `#include` .*
- class [list](#)  
*The STL list container class. Use with `#include` .*
- class [slist](#)  
*The STL slist container class. Use with `#include` .*
- class [set](#)  
*The STL set container class. Use with `#include` .*
- class [map](#)  
*The STL map container class. Use with `#include` .*
- class [multiset](#)  
*The STL multiset container class. Use with `#include` .*
- class [multimap](#)  
*The STL multimap container class. Use with `#include` .*
- class [string](#)  
*The STL string class. Use with `#include` .*

### 9.3.1 Detailed Description

The Standard Template Library (STL).

This is not the real STL documentation, you can find that at the [Silicon Graphics STL site](#).

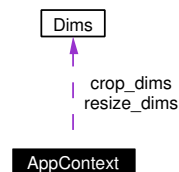
## Chapter 10

# iLab Neuromorphic Vision Toolkit Class Documentation

### 10.1 AppContext Class Reference

Manages state information for vision.C::main().

Collaboration diagram for AppContext:



#### Public Member Functions

- void **normalizeSalmap** ([Image](#)< float > &salmap) const
- template<class T> [Image](#)< T > **resizeImage** (const [Image](#)< T > &input) const
- bool **isCropping** () const
- template<class T> [Image](#)< T > **foaCrop** (const [Image](#)< T > &input, const [Point2D](#) &foa)
- void **waitKbd** ()
- void **saveCS** (const [SingleChannel](#) &chan, const char \*filestem, const int frame\_nr=-1, const char \*img\_name=NULL, bool justsave=false)

- void **saveChannelOutput** ([ChannelBase](#) &chan, const char \*name, const int frame\_nr=-1, const char \*img\_name=NULL, bool justsave=false)
- void **saveLandmark** (const [SingleChannel](#) &chan, const char \*filestem, const int frame\_nr=-1, const char \*img\_name=NULL, bool justsave=false)

## Public Attributes

- float **salmap\_factor**
- int **qlen**
- [Dims](#) **resize\_dims**
- [Dims](#) **crop\_dims**
- bool **is\_interactive**
- int **nborient**
- const char \* **sox\_channel**
- const char \* **contour\_channel**
- int **exit\_after\_display**
- int **embed**
- bool **saveShapeEstimator**
- [std::vector](#)< [std::string](#) > **save\_submaps**
- [std::vector](#)< [std::string](#) > **save\_output**

### 10.1.1 Detailed Description

Manages state information for `vision.C::main()`.

It holds state variables that can be controlled by command-line options, and it has little helper functions to wrap functionality that is used several times in `main()`.

Definition at line 178 of file `vision.C`.

The documentation for this class was generated from the following file:

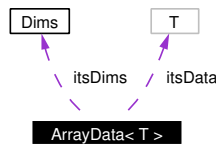
- [vision.C](#)

## 10.2 `ArrayData< T >` Class Template Reference

Implements a ref-counted 2-D array, should only be used via [ArrayHandle](#).

```
#include <ArrayData.H>
```

Collaboration diagram for `ArrayData< T >`:



### Public Member Functions

- [ArrayData](#) ()  
*Default construct an empty (i.e. 0-by-0) array.*
- [ArrayData](#) (const [Dims](#) &dims, const [InitPolicy](#) ip=NO\_INIT)  
*Construct with the specified dimensions and InitPolicy.*
- [ArrayData](#) (const [Dims](#) &dims, const T \*aa)  
*Construct with the specified dimensions.*
- [ArrayData](#) (const [Dims](#) &dims, T \*aa, const [StoragePolicy](#) s)  
*Construct with the specified dimensions and storage policy.*
- [ArrayData](#) \* [clone](#) () const  
*Return a new'ed copy of \*this.*
- [~ArrayData](#) ()  
*Release the pointed-to data if our storage policy is MAKE\_OWN\_COPY.*
- const T \* [data](#) () const throw ()  
*Returns a read-only pointer to the data array.*
- const T \* [end](#) () const throw ()  
*Returns a read-only pointer to one-past-the-end of the data array.*
- T \* [dataw](#) () throw ()  
*Returns a read-write pointer to the data array (the "w" means "write").*

- `T * endw () throw ()`  
Returns a read-write pointer to one-past-the-end of the data array.
- `const Dims & dims () const throw ()`  
Returns the dimensions (i.e. width, height) of the 2-D data array.
- `int w () const throw ()`  
Returns the width of the data array (i.e. the number of columns).
- `int h () const throw ()`  
Returns the height of the data array (i.e. the number of rows).
- `void acquire () throw ()`  
Increment the reference count.
- `void release () throw ()`  
Decrement the reference count and delete self if the count goes to zero.
- `bool isShared () const throw ()`  
Query whether this object is shared (i.e. ref count > 1).
- `int refCount () const throw ()`  
For debugging/testing only.

### 10.2.1 Detailed Description

**template<class T> class ArrayData< T >**

Implements a ref-counted 2-D array, should only be used via [ArrayHandle](#).

ArrayData implements a 2-D array, by storing:

- a [Dims](#) representing the width+height of the array,
- a pointer to the dynamically allocated memory where the array data are stored,
- a reference count, and
- a StoragePolicy, indicating whether the ArrayData object "owns" the memory and should delete[] it eventually (in the case of MAKE\_OWN\_COPY), or whether someone else "owns" the memory and will eventually delete[] it themselves (in the case of WRITE\_THRU)

All of these data members are immutable, with the exception of the reference count. This reflects the semantics that an `ArrayData` object is a fixed-size block of data; if a higher-level clients wants to `resize()`, it does so by creating a new `ArrayData` of the desired size, and releasing the old `ArrayData`. This simplifies the memory management: within `ArrayData`, memory is `new[]`'ed only in the constructor(s), and `delete[]`'ed only in the destructor.

`ArrayData` provides the basics needed to use STL-style iterator algorithms. In particular it offers two pairs of functions to get half-open iterator ranges:

- `data()` and `end()` for read-only iteration, and
- `dataw()` and `endw()` for read-write iteration (where the "w" means "write" or "writable").

Definition at line 141 of file `ArrayData.H`.

## 10.2.2 Constructor & Destructor Documentation

### 10.2.2.1 `template<class T> ArrayData< T >::ArrayData ()` [inline]

Default construct an empty (i.e. 0-by-0) array.

Definition at line 420 of file `ArrayData.H`.

References `MAKE_OWN_COPY`, and `RC_SET`.

Referenced by `ArrayData< T >::clone()`.

### 10.2.2.2 `template<class T> ArrayData< T >::ArrayData (const Dims & dims, const InitPolicy ip = NO_INIT)` [inline]

Construct with the specified dimensions and `InitPolicy`.

Definition at line 435 of file `ArrayData.H`.

References `MAKE_OWN_COPY`, `ArrayHelper< T, isTrivial >::minimal_initialize()`, `RC_SET`, `Dims::sz()`, `ArrayHelper< T, isTrivial >::zero_initialize()`, and `ZEROS`.

### 10.2.2.3 `template<class T> ArrayData< T >::ArrayData (const Dims & dims, const T * aa)` [inline]

Construct with the specified dimensions.

If the passed-in `T*` array is non-null, copy that data into the constructed object.

Definition at line 455 of file `ArrayData.H`.

References `ArrayHelper< T, isTrivial >::copy_initialize()`, `MAKE_OWN_COPY`, `RC_SET`, and `Dims::sz()`.

#### 10.2.2.4 `template<class T> ArrayData< T >::ArrayData (const Dims & dims, T * aa, const StoragePolicy s) [inline]`

Construct with the specified dimensions and storage policy.

If the passed-in T\* array is non-null, then if the storage policy is `MAKE_OWN_COPY` we make a copy of the data, but if the storage policy is `WRITE_THRU` then we "borrow" the data.

Definition at line 472 of file `ArrayData.H`.

References `ArrayHelper< T, isTrivial >::copy_initialize()`, `MAKE_OWN_COPY`, `ArrayHelper< T, isTrivial >::minimal_initialize()`, `RC_SET`, and `Dims::sz()`.

#### 10.2.2.5 `template<class T> ArrayData< T >::~ArrayData () [inline]`

Release the pointed-to data if our storage policy is `MAKE_OWN_COPY`.

Definition at line 508 of file `ArrayData.H`.

References `ArrayHelper< T, isTrivial >::destruct()`, `MAKE_OWN_COPY`, and `Dims::sz()`.

### 10.2.3 Member Function Documentation

#### 10.2.3.1 `template<class T> void ArrayData< T >::acquire () throw () [inline]`

Increment the reference count.

Definition at line 557 of file `ArrayData.H`.

References `RC_INCR`.

#### 10.2.3.2 `template<class T> ArrayData< T > * ArrayData< T >::clone () const [inline]`

Return a new'ed copy of \*this.

The cloned copy will always be `MAKE_OWN_COPY`, even if \*this is `WRITE_THRU`.

Definition at line 497 of file `ArrayData.H`.

References `ArrayData< T >::ArrayData()`.



**10.2.3.3** `template<class T> const T * ArrayData< T >::data () const throw ()`  
[inline]

Returns a read-only pointer to the data array.

With `end()`, forms an STL-style half-open const\_iterator range.

Definition at line 522 of file ArrayData.H.

**10.2.3.4** `template<class T> T * ArrayData< T >::dataw () throw ()`  
[inline]

Returns a read-write pointer to the data array (the "w" means "write").

With `endw()`, forms an STL-style half-open iterator range.

Definition at line 532 of file ArrayData.H.

**10.2.3.5** `template<class T> const Dims & ArrayData< T >::dims () const`  
`throw ()` [inline]

Returns the dimensions (i.e. width, height) of the 2-D data array.

Definition at line 542 of file ArrayData.H.

**10.2.3.6** `template<class T> const T * ArrayData< T >::end () const throw ()`  
[inline]

Returns a read-only pointer to one-past-the-end of the data array.

With `data()`, forms an STL-style half-open const\_iterator range.

Definition at line 527 of file ArrayData.H.

**10.2.3.7** `template<class T> T * ArrayData< T >::endw () throw ()`  
[inline]

Returns a read-write pointer to one-past-the-end of the data array.

With `dataw()`, forms an STL-style half-open iterator range.

Definition at line 537 of file ArrayData.H.

**10.2.3.8** `template<class T> int ArrayData< T >::h () const throw ()`  
[inline]

Returns the height of the data array (i.e. the number of rows).

Definition at line 552 of file ArrayData.H.

**10.2.3.9** `template<class T> bool ArrayData< T >::isShared () const throw ()  
[inline]`

Query whether this object is shared (i.e. ref count > 1).

Definition at line 571 of file ArrayData.H.

References RC\_GET.

**10.2.3.10** `template<class T> int ArrayData< T >::refCount () const throw ()  
[inline]`

For debugging/testing only.

Returns the current reference count.

Definition at line 578 of file ArrayData.H.

References RC\_GET.

**10.2.3.11** `template<class T> void ArrayData< T >::release () throw ()  
[inline]`

Decrement the reference count and delete self if the count goes to zero.

Definition at line 564 of file ArrayData.H.

References RC\_DECR\_TEST\_ZERO.

**10.2.3.12** `template<class T> int ArrayData< T >::w () const throw ()  
[inline]`

Returns the width of the data array (i.e. the number of columns).

Definition at line 547 of file ArrayData.H.

The documentation for this class was generated from the following file:

- [ArrayData.H](#)

## 10.3 ArrayHandle< T > Class Template Reference

This class provides ref-counting and copy-on-write for ArrayData's.

```
#include <ArrayData.H>
```

### Public Member Functions

- [ArrayHandle \(\)](#)  
*Default construct with an empty array (i.e. 0-by-0).*
- [ArrayHandle \(const ArrayData< T > \\*p\) throw \(\)](#)  
*Construct with a new [ArrayData](#), which the ArrayHandle now owns.*
- [ArrayHandle \(const ArrayHandle &r\) throw \(\)](#)  
*Copy construct; no deep-copy is incurred, just bumps the ref-count.*
- [~ArrayHandle \(\) throw \(\)](#)  
*Destruct; delete's the ArrayData<T> if its ref count goes to zero.*
- void [swap \(ArrayHandle< T > &other\) throw \(\)](#)  
*Swap contents with another ArrayHandle.*
- [ArrayHandle & operator= \(const ArrayHandle &r\) throw \(\)](#)  
*Assignment; no deep-copy is incurred, just bumps the ref-count.*
- const [ArrayData< T > & get \(\) const throw \(\)](#)  
*Get a read-only reference to the [ArrayData](#) object.*
- [ArrayData< T > & uniq \(\)](#)  
*Get a unique read-write reference to the [ArrayData](#) object.*
- bool [isShared \(\) const throw \(\)](#)  
*See if the [ArrayData](#) object is shared (i.e. ref-count > 1).*
- bool [hasSameData \(const ArrayHandle< T > &b\) const throw \(\)](#)  
*For testing/debugging only.*
- int [refCount \(\) const throw \(\)](#)  
*For testing/debugging only.*

### 10.3.1 Detailed Description

**template<class T> class ArrayHandle< T >**

This class provides ref-counting and copy-on-write for ArrayData's.

NOTE: Most clients will not need to use ArrayHandle<T> directly, but can rather use Image<T>, which wraps and expands the ArrayHandle interface. Nevertheless, these notes may be of interest because the ref-counting and copy-on-write semantics that ArrayHandle<T> offers are transferred through to Image<T>.

As long as clients access ArrayData<T> through an ArrayHandle<T>, the proper copy-on-write semantics are guaranteed. This is by virtue of the fact that the only way to extract a non-const ArrayData<T>& from an ArrayHandle<T> is to call `uniq()`, which in turn checks to see if the [ArrayData](#) is shared (i.e. ref count > 1). If it is shared, then that [ArrayData](#) is clone()'d before returning a reference to it.

The only caveat: you should not store non-const references to an [ArrayData](#) or to the elements of an [ArrayData](#) object for "long" periods of time. In this case, "long" means "longer than you are sure you have a unique reference", which in practice means "you can keep such references up until a new copy of the ArrayHandle<T> is made". This goes along with sensible coding practices anyway; a typical usage pattern would be:

1. you have an ArrayHandle<T> (or an Image<T>, which holds an ArrayHandle<T>) that you want to modify
2. get a pair of iterator's for that object so that you can perform your series of write operations
3. don't use those iterators anymore, and then you can safely make copies of the freshly-modified ArrayHandle<T> or Image<T> object (for example to return a copy of it as the return value from a function)

Definition at line 263 of file ArrayData.H.

### 10.3.2 Constructor & Destructor Documentation

**10.3.2.1 template<class T> [ArrayHandle< T >::ArrayHandle](#) () [inline]**

Default construct with an empty array (i.e. 0-by-0).

Definition at line 591 of file ArrayData.H.

**10.3.2.2 template<class T> [ArrayHandle< T >::ArrayHandle](#) (const [ArrayData< T > \\*p](#)) throw () [inline, explicit]**

Construct with a new [ArrayData](#), which the ArrayHandle now owns.

Definition at line 598 of file ArrayData.H.

**10.3.2.3** `template<class T> ArrayHandle< T >::ArrayHandle (const  
ArrayHandle< T > & r) throw () [inline]`

Copy construct; no deep-copy is incurred, just bumps the ref-count.

Definition at line 608 of file ArrayData.H.

**10.3.2.4** `template<class T> ArrayHandle< T >::~~ArrayHandle () throw ()  
[inline]`

Destruct; delete's the ArrayData<T> if its ref count goes to zero.

Definition at line 616 of file ArrayData.H.

### 10.3.3 Member Function Documentation

**10.3.3.1** `template<class T> const ArrayData< T > & ArrayHandle< T >::get  
() const throw () [inline]`

Get a read-only reference to the [ArrayData](#) object.

This is a "cheap" operation, since the [ArrayData](#) will never be copied in this operation.

Definition at line 639 of file ArrayData.H.

**10.3.3.2** `template<class T> bool ArrayHandle< T >::hasSameData (const  
ArrayHandle< T > & b) const throw () [inline]`

For testing/debugging only.

See if we are pointing to the same ArrayData<T> as is the other ArrayHandle<T>.

Definition at line 662 of file ArrayData.H.

**10.3.3.3** `template<class T> bool ArrayHandle< T >::isShared () const throw  
() [inline]`

See if the [ArrayData](#) object is shared (i.e. ref-count > 1).

Definition at line 657 of file ArrayData.H.

**10.3.3.4** `template<class T> ArrayHandle< T > & ArrayHandle< T >::operator= (const ArrayHandle< T > & r) throw () [inline]`

Assignment; no deep-copy is incurred, just bumps the ref-count.

Definition at line 632 of file `ArrayData.H`.

**10.3.3.5** `template<class T> int ArrayHandle< T >::refCount () const throw () [inline]`

For testing/debugging only.

Returns the current reference count of the our `ArrayData<T>`.

Definition at line 667 of file `ArrayData.H`.

**10.3.3.6** `template<class T> void ArrayHandle< T >::swap (ArrayHandle< T > & other) throw () [inline]`

Swap contents with another `ArrayHandle`.

Definition at line 623 of file `ArrayData.H`.

Referenced by `ArrayHandle< T >::uniq()`.

**10.3.3.7** `template<class T> ArrayData< T > & ArrayHandle< T >::uniq () [inline]`

Get a unique read-write reference to the `ArrayData` object.

This is a potentially "expensive" operation, since in order to get a unique `ArrayData` object, the current one may have to be clone()'d if it is shared (i.e. ref-count > 1).

Definition at line 644 of file `ArrayData.H`.

References `copy()`, and `ArrayHandle< T >::swap()`.

The documentation for this class was generated from the following file:

- [ArrayData.H](#)

## 10.4 ArrayHelper< T, isTrivial > Struct Template Reference

This is a helper template struct for ArrayData's constructors.

```
#include <ArrayData.H>
```

### Static Public Member Functions

- void `minimal_initialize` (T \*space, int n)  
*Construct n copies of val at space using placement new.*
- void `zero_initialize` (T \*space, int n)  
*Initialize the memory to T's default or zero value.*
- void `copy_initialize` (T \*space, int n, const T \*src)  
*Initialize the memory by copying elements from the src array.*
- void `destruct` (T \*space, int n)  
*Call destructors on the n objects stored in space.*

### 10.4.1 Detailed Description

```
template<class T, bool isTrivial = TypeTraits<T>::isTrivial> struct Array-  
Helper< T, isTrivial >
```

This is a helper template struct for ArrayData's constructors.

It exposes functions for initializing raw memory in different ways. These functions can then be specialized for builtin or trivial types (i.e., types for which TypeTraits<T>::isTrivial is true) for maximum efficiency. We get this specialization by having an extra bool isTrivial template parameter for ArrayHelper, which allows us to select at compile-time based on the complexity of T. This is much better than using a run-time "if" statement in the constructors, since that incurs both a time cost (to evaluate an "if" statement) and a code space cost (since we're then including code that won't be used).

Definition at line 330 of file ArrayData.H.

## 10.4.2 Member Function Documentation

**10.4.2.1** `template<class T, bool isTrivial = TypeTraits<T>::isTrivial> void  
ArrayHelper< T, isTrivial >::copy_initialize (T * space, int n, const T  
 * src) [inline, static]`

Initialize the memory by copying elements from the *src* array.

For general types, we just copy construct each element in the target array from the corresponding element in the *src* array, using placement new.

Definition at line 367 of file `ArrayData.H`.

Referenced by `ArrayData< T >::ArrayData()`.

**10.4.2.2** `template<class T, bool isTrivial = TypeTraits<T>::isTrivial> void  
ArrayHelper< T, isTrivial >::destruct (T * space, int n) [inline,  
 static]`

Call destructors on the *n* objects stored in *space*.

Definition at line 385 of file `ArrayData.H`.

Referenced by `ArrayData< T >::~~ArrayData()`.

**10.4.2.3** `template<class T, bool isTrivial = TypeTraits<T>::isTrivial> void  
ArrayHelper< T, isTrivial >::minimal_initialize (T * space, int n)  
 [inline, static]`

Construct *n* copies of *val* at *space* using placement new.

For general types, the minimal initialization that we can do is to `zero_initialize()` (i.e., unlike for builtin or trivial types, we can't just leave the memory sitting as it is, uninitialized).

Definition at line 336 of file `ArrayData.H`.

References `ArrayHelper< T, isTrivial >::zero_initialize()`.

Referenced by `ArrayData< T >::ArrayData()`.

**10.4.2.4** `template<class T, bool isTrivial = TypeTraits<T>::isTrivial> void  
ArrayHelper< T, isTrivial >::zero_initialize (T * space, int n)  
 [inline, static]`

Initialize the memory to T's default or zero value.

For general types, the best way to do this is to make one default T object, and then copy construct each element of the array from that object using placement new.



Definition at line 345 of file ArrayData.H.

Referenced by `ArrayData< T >::ArrayData()`, and `ArrayHelper< T, isTrivial >::minimal_initialize()`.

The documentation for this struct was generated from the following file:

- [ArrayData.H](#)

## 10.5 `ArrayHelper< T, true >` Struct Template Reference

Specialization of [ArrayHelper](#) for builtin/trivial types.

```
#include <ArrayData.H>
```

### Static Public Member Functions

- void [minimal\\_initialize](#) (T \*, int)  
*For trivial types, we don't have to do anything here.*
- void [zero\\_initialize](#) (T \*space, int n)  
*For trivial types, we just memset() all the memory to zero.*
- void [copy\\_initialize](#) (T \*space, int n, const T \*src)  
*For trivial types, we just memcpy() into the destination.*
- void [destruct](#) (T \*, int)  
*For trivial types, no destruction is needed.*

### 10.5.1 Detailed Description

```
template<class T> struct ArrayHelper< T, true >
```

Specialization of [ArrayHelper](#) for builtin/trivial types.

We can do things much faster with these kinds of types, using e.g. `memset()` and `memcpy()`, and we don't have to do default initialization at all.

Definition at line 397 of file `ArrayData.H`.

### 10.5.2 Member Function Documentation

**10.5.2.1** `template<class T> void ArrayHelper< T, true >::copy_initialize (T *space, int n, const T *src) [inline, static]`

For trivial types, we just `memcpy()` into the destination.

Definition at line 409 of file `ArrayData.H`.

**10.5.2.2** `template<class T> void ArrayHelper< T, true >::destruct (T *, int) [inline, static]`

For trivial types, no destruction is needed.

Definition at line 415 of file ArrayData.H.

**10.5.2.3** `template<class T> void ArrayHelper< T, true >::minimal_initialize (T *, int) [inline, static]`

For trivial types, we don't have to do anything here.

Definition at line 400 of file ArrayData.H.

**10.5.2.4** `template<class T> void ArrayHelper< T, true >::zero_initialize (T * space, int n) [inline, static]`

For trivial types, we just memset() all the memory to zero.

Definition at line 403 of file ArrayData.H.

The documentation for this struct was generated from the following file:

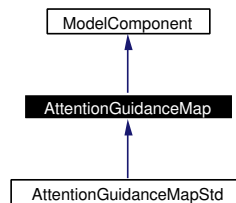
- [ArrayData.H](#)

## 10.6 AttentionGuidanceMap Class Reference

The task-relevance map base class.

```
#include <AttentionGuidanceMap.H>
```

Inheritance diagram for AttentionGuidanceMap:



Collaboration diagram for AttentionGuidanceMap:

### Public Member Functions

#### Constructor, destructor, and reset

- [AttentionGuidanceMap](#) ([ModelManager](#) &mgr, const [std::string](#) &descr-Name="Attention Guidance Map", const [std::string](#) &tagName="Attention-GuidanceMap")  
*Initialized constructor.*
- virtual void [exportOptions](#) (const int whatToExport, const bool recurse=true)  
*Export some of our [ModelParam](#) parameters as command-line-options.*
- virtual [~AttentionGuidanceMap](#) ()  
*Destructor.*
- virtual void [reset](#) ()=0  
*Reset to initial state just after construction.*
- void [setBrain](#) (const [SharedPtr](#)< [Brain](#) > &brain)  
*Setup a hook back to our brain.*

#### Input, evolve and output

- virtual void [inputBU](#) (const [Image](#)< float > &in)=0  
*Set new bottom-up input.*

- virtual void `inputTD` (const `Image< float >` &in)=0  
*Set new top-down input.*
- virtual int `getWidth` () const=0  
*Get width of the map.*
- virtual int `getHeight` () const=0  
*Get height of the map.*
- virtual `Image< float >` `getV` (const bool normalize=false) const=0  
*Return all our relevance values as an `Image<float>`.*
- virtual float `getV` (const `Point2D` &p) const=0  
*Access function using `Point2D`.*
- virtual void `evolve` (const double t)=0  
*Integrate inputs until time t (in s) and update internal state.*

## Results

- virtual void `saveResults` (const `SharedPtr< OutputFrameSeries >` &ofs)  
*Save our various results.*

## Protected Attributes

- `ModelParam< bool >` `itsSaveResults`  
*Save our internals when `saveResults()` is called?*
- `ModelParam< float >` `itsAGMFactor`  
*Multiply by a factor rather than normalize when doing `getV()`?*
- `SharedPtr< Brain >` `itsBrain`  
*Our brain, or `SharedPtr<NULL>` if it has not been set with `setBrain()`.*

### 10.6.1 Detailed Description

The task-relevance map base class.

This is a 2D task-relevance map. It is just a base class with virtual function definitions. Various implementations are available below. The TRM is a topographic map where a

value of zero indicates that a location is not relevant to the task at hand, while a value of 1.0 indicates neutral relevance and values higher than 1.0 indicate high relevance.

Definition at line 55 of file AttentionGuidanceMap.H.

## 10.6.2 Constructor & Destructor Documentation

### 10.6.2.1 AttentionGuidanceMap::AttentionGuidanceMap ([ModelManager](#) & *mgr*, const [std::string](#) & *descrName* = "Attention Guidance Map", const [std::string](#) & *tagName* = "AttentionGuidanceMap")

Initialized constructor.

The map will be resized and initialized the first time input() is called

Definition at line 55 of file AttentionGuidanceMap.C.

### 10.6.2.2 AttentionGuidanceMap::~AttentionGuidanceMap () [virtual]

Destructor.

Definition at line 65 of file AttentionGuidanceMap.C.

## 10.6.3 Member Function Documentation

### 10.6.3.1 virtual void AttentionGuidanceMap::evolve (const double *t*) [pure virtual]

Integrate inputs until time *t* (in s) and update internal state.

Implemented in [AttentionGuidanceMapStd](#).

### 10.6.3.2 void AttentionGuidanceMap::exportOptions (const int *whatToExport*, const bool *recurse* = true) [virtual]

Export some of our [ModelParam](#) parameters as command-line-options.

See the base function in [ModelComponent.H](#) for info. A base implementation is provided that will only export the base class options (see the [ModelParam](#) definitions in the protected section below).

Reimplemented from [ModelComponent](#).

Definition at line 69 of file AttentionGuidanceMap.C.

References [ModelComponent::exportOptions\(\)](#), [itsAGMFactor](#), [itsSaveResults](#), [OPTEXP\\_CORE](#), [OPTEXP\\_SAVE](#), and [ModelManager::requestOption\(\)](#).

**10.6.3.3** `virtual int AttentionGuidanceMap::getHeight () const` [pure virtual]

Get height of the map.

Implemented in [AttentionGuidanceMapStd](#).

**10.6.3.4** `virtual float AttentionGuidanceMap::getV (const Point2D & p) const` [pure virtual]

Access function using [Point2D](#).

Implemented in [AttentionGuidanceMapStd](#).

**10.6.3.5** `virtual Image<float> AttentionGuidanceMap::getV (const bool normalize = false) const` [pure virtual]

Return all our relevance values as an [Image](#)<float>.

The current values are possibly normalized to 0..255. The normalization will use its-TRMFactor if it is non-null

Implemented in [AttentionGuidanceMapStd](#).

Referenced by `saveResults()`.

**10.6.3.6** `virtual int AttentionGuidanceMap::getWidth () const` [pure virtual]

Get width of the map.

Implemented in [AttentionGuidanceMapStd](#).

**10.6.3.7** `virtual void AttentionGuidanceMap::inputBU (const Image< float > & in) [pure virtual]`

Set new bottom-up input.

This should initialize and resize the map if it is currently uninitialized (e.g., just after construction or [reset\(\)](#)).

Implemented in [AttentionGuidanceMapStd](#).

**10.6.3.8 virtual void AttentionGuidanceMap::inputTD (const [Image](#)< float > & in) [pure virtual]**

Set new top-down input.

This should initialize and resize the map if it is currently uninitialized (e.g., just after construction or [reset\(\)](#)).

Implemented in [AttentionGuidanceMapStd](#).

**10.6.3.9 virtual void AttentionGuidanceMap::reset () [pure virtual]**

Reset to initial state just after construction.

Implemented in [AttentionGuidanceMapStd](#).

**10.6.3.10 void AttentionGuidanceMap::saveResults (const [SharedPtr](#)< [OutputFrameSeries](#) > & ofs) [virtual]**

Save our various results.

A default implementation is provided, which just calls [getV\(\)](#) and saves that map with a file name prefix of "SM".

**Parameters:**

*ofs* will figure out file name/format and save our results.

Definition at line 84 of file [AttentionGuidanceMap.C](#).

References [getV\(\)](#), [ModelParam](#)< bool >::getVal(), and [itsSaveResults](#).

**10.6.3.11 void AttentionGuidanceMap::setBrain (const [SharedPtr](#)< [Brain](#) > & brain)**

Setup a hook back to our brain.

Definition at line 91 of file [AttentionGuidanceMap.C](#).

References [itsBrain](#).

**10.6.4 Member Data Documentation****10.6.4.1 [ModelParam](#)<float> [AttentionGuidanceMap::itsAGMFactor](#) [protected]**

Multiply by a factor rather than normalize when doing [getV\(\)](#)?



Definition at line 139 of file AttentionGuidanceMap.H.

Referenced by exportOptions().

#### 10.6.4.2 [SharedPtr<Brain>](#) [AttentionGuidanceMap::itsBrain](#) [protected]

Our brain, or SharedPtr<NULL> if it has not been set with [setBrain\(\)](#).

Definition at line 142 of file AttentionGuidanceMap.H.

Referenced by setBrain().

#### 10.6.4.3 [ModelParam<bool>](#) [AttentionGuidanceMap::itsSaveResults](#) [protected]

Save our internals when [saveResults\(\)](#) is called?

Definition at line 136 of file AttentionGuidanceMap.H.

Referenced by exportOptions(), and saveResults().

The documentation for this class was generated from the following files:

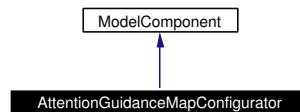
- [AttentionGuidanceMap.H](#)
- [AttentionGuidanceMap.C](#)

## 10.7 AttentionGuidanceMapConfigurator Class Reference

[AttentionGuidanceMap](#) configurator.

```
#include <AttentionGuidanceMap.H>
```

Inheritance diagram for AttentionGuidanceMapConfigurator:



Collaboration diagram for AttentionGuidanceMapConfigurator:

### Public Member Functions

- [AttentionGuidanceMapConfigurator](#) ([ModelManager](#) &mgr, const [std::string](#) &descrName="Attention Guidance Map Configurator", const [std::string](#) &tagName="AttentionGuidanceMapConfigurator")

*Constructor.*

- virtual [~AttentionGuidanceMapConfigurator](#) ()

*destructor*

- virtual void [exportOptions](#) (const int whatToExport, const bool recurse=true)

*Export some of our [ModelParam](#) parameters as command-line-options.*

- [SharedPtr](#)< [AttentionGuidanceMap](#) > [getAGM](#) () const

*Get the chosen AGM.*

### Protected Member Functions

- virtual bool [setModelParamString](#) (const [std::string](#) &name, const [std::string](#) &value, const bool recurse=false, const bool warn=true)

*Intercept people changing our [ModelParam](#).*

## Protected Attributes

- [ModelParam](#)< [std::string](#) > [itsAGMtype](#)  
*type of map*

### 10.7.1 Detailed Description

[AttentionGuidanceMap](#) configurator.

This will export the `-agm-type=XX` command-line option and will instantiate an AGM of the desired type as the option gets assigned a value. As this happens, new options may become available in the command-line. To see them, use `-help` AFTER you have chosen the type to use. The current AGM may be retrieved using [getAGM\(\)](#).

Definition at line 158 of file [AttentionGuidanceMap.H](#).

### 10.7.2 Constructor & Destructor Documentation

**10.7.2.1** [AttentionGuidanceMapConfigurator::AttentionGuidanceMapConfigurator](#) ([ModelManager](#) & *mgr*, const [std::string](#) & *descrName* = "Attention Guidance Map Configurator", const [std::string](#) & *tagName* = "AttentionGuidanceMapConfigurator")

Constructor.

Definition at line 100 of file [AttentionGuidanceMap.C](#).

References [OPTEXP\\_NONE](#).

**10.7.2.2** [AttentionGuidanceMapConfigurator::~~AttentionGuidanceMapConfigurator](#) () [[virtual](#)]

destructor

Definition at line 109 of file [AttentionGuidanceMap.C](#).

### 10.7.3 Member Function Documentation

**10.7.3.1** [void AttentionGuidanceMapConfigurator::exportOptions](#) (const int *whatToExport*, const bool *recurse* = true) [[virtual](#)]

Export some of our [ModelParam](#) parameters as command-line-options.

See the base function in [ModelComponent.H](#) for info. Actually, here we just keep a copy of whatToExport and we will call [exportOptions\(\)](#) on our SM once we know its type and instantiate it.

Reimplemented from [ModelComponent](#).

Definition at line 113 of file AttentionGuidanceMap.C.

References [ModelComponent::exportOptions\(\)](#), [itsAGMtype](#), [OPTEXP\\_CORE](#), and [ModelManager::requestOption\(\)](#).

#### 10.7.3.2 [SharedPtr](#)< [AttentionGuidanceMap](#) > [AttentionGuidanceMapConfigurator::getAGM \(\) const](#)

Get the chosen AGM.

You should call this during [start\(\)](#) of the [ModelComponent](#) that needs the AGM. CAUTION: this may be a [SharedPtr\(NULL\)](#) if no AGM type has been chosen.

Definition at line 130 of file AttentionGuidanceMap.C.

#### 10.7.3.3 **bool** [AttentionGuidanceMapConfigurator::setModelParamString](#) (const [std::string](#) & name, const [std::string](#) & value, const bool recurse = false, const bool warn = true) [[protected](#), [virtual](#)]

Intercept people changing our [ModelParam](#).

See [ModelComponent.H](#); as parsing the command-line or reading a config file sets our name, we'll also here instantiate a controller of the proper type (and export its options)

Reimplemented from [ModelComponent](#).

Definition at line 135 of file AttentionGuidanceMap.C.

References [ModelComponent::addSubComponent\(\)](#), [SharedPtr< AttentionGuidanceMap >::get\(\)](#), [itsAGMtype](#), [ModelComponent::removeSubComponent\(\)](#), [SharedPtr< AttentionGuidanceMap >::reset\(\)](#), and [ModelComponent::setModelParamString\(\)](#).

### 10.7.4 Member Data Documentation

#### 10.7.4.1 [ModelParam](#)<[std::string](#)> [AttentionGuidanceMapConfigurator::itsAGMtype](#) [[protected](#)]

type of map

Definition at line 186 of file AttentionGuidanceMap.H.

Referenced by [exportOptions\(\)](#), and [setModelParamString\(\)](#).

The documentation for this class was generated from the following files:

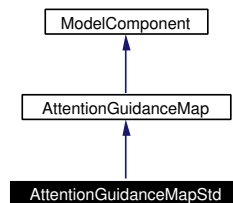
- [AttentionGuidanceMap.H](#)
- [AttentionGuidanceMap.C](#)

## 10.8 AttentionGuidanceMapStd Class Reference

The standard task-relevance map.

```
#include <AttentionGuidanceMap.H>
```

Inheritance diagram for AttentionGuidanceMapStd:



Collaboration diagram for AttentionGuidanceMapStd:

### Public Member Functions

- [AttentionGuidanceMapStd](#) ([ModelManager](#) &mgr, const [std::string](#) &descr-Name="Task-Relevance Map Std", const [std::string](#) &tagName="Attention-GuidanceMapStd")

*Uninitialized constructor.*

- virtual [~AttentionGuidanceMapStd](#) ()

*Destructor.*

- virtual void [reset](#) ()

*Reset to initial state just after construction.*

- virtual void [inputBU](#) (const [Image](#)< float > &in)

*Set new bottom-up input.*

- virtual void [inputTD](#) (const [Image](#)< float > &in)

*Set new top-down input.*

- virtual int [getWidth](#) () const

*Get width of the map.*

- virtual int [getHeight](#) () const

*Get height of the map.*

- virtual `Image< float > getV (const bool normalize=false) const`  
*Return all our relevance values as an `Image<float>`.*
- virtual `float getV (const Point2D &p) const`  
*Access function using `Point2D`.*
- virtual `void evolve (const double t)`  
*Integrate inputs until time  $t$  (in s) and update internal state.*

### 10.8.1 Detailed Description

The standard task-relevance map.

This is our current standard AGM implementation. It just takes the pointwise product between bottom-up and top-down inputs.

Definition at line 208 of file AttentionGuidanceMap.H.

### 10.8.2 Constructor & Destructor Documentation

**10.8.2.1** `AttentionGuidanceMapStd::AttentionGuidanceMapStd (ModelManager & mgr, const std::string & descrName = "Task-Relevance Map Std", const std::string & tagName = "AttentionGuidanceMapStd")`

Uninitialized constructor.

Definition at line 182 of file AttentionGuidanceMap.C.

**10.8.2.2** `AttentionGuidanceMapStd::~AttentionGuidanceMapStd () [virtual]`

Destructor.

Definition at line 189 of file AttentionGuidanceMap.C.

### 10.8.3 Member Function Documentation

**10.8.3.1** `void AttentionGuidanceMapStd::evolve (const double t) [virtual]`

Integrate inputs until time  $t$  (in s) and update internal state.

Implements [AttentionGuidanceMap](#).

Definition at line 221 of file AttentionGuidanceMap.C.

#### 10.8.3.2 `int AttentionGuidanceMapStd::getHeight () const` [virtual]

Get height of the map.

Implements [AttentionGuidanceMap](#).

Definition at line 233 of file AttentionGuidanceMap.C.

References `Image< float >::getHeight()`.

#### 10.8.3.3 `float AttentionGuidanceMapStd::getV (const Point2D & p) const` [virtual]

Access function using [Point2D](#).

Implements [AttentionGuidanceMap](#).

Definition at line 225 of file AttentionGuidanceMap.C.

References `Image< float >::getVal()`.

#### 10.8.3.4 `Image< float > AttentionGuidanceMapStd::getV (const bool normalize = false) const` [virtual]

Return all our relevance values as an [Image<float>](#).

Implements [AttentionGuidanceMap](#).

Definition at line 205 of file AttentionGuidanceMap.C.

References `Image< float >::getHeight()`, `ModelParam< float >::getVal()`, `Image< float >::getWidth()`, `Image< float >::isSameSize()`, and `Image< T >::normalize()`.

#### 10.8.3.5 `int AttentionGuidanceMapStd::getWidth () const` [virtual]

Get width of the map.

Implements [AttentionGuidanceMap](#).

Definition at line 229 of file AttentionGuidanceMap.C.

References `Image< float >::getWidth()`.



**10.8.3.6 void AttentionGuidanceMapStd::inputBU (const [Image](#)< float > & in)**  
[virtual]

Set new bottom-up input.

Implements [AttentionGuidanceMap](#).

Definition at line 197 of file AttentionGuidanceMap.C.

**10.8.3.7 void AttentionGuidanceMapStd::inputTD (const [Image](#)< float > & in)**  
[virtual]

Set new top-down input.

Implements [AttentionGuidanceMap](#).

Definition at line 201 of file AttentionGuidanceMap.C.

**10.8.3.8 void AttentionGuidanceMapStd::reset ()** [virtual]

Reset to initial state just after construction.

Implements [AttentionGuidanceMap](#).

Definition at line 193 of file AttentionGuidanceMap.C.

References [Image](#)< float >::freeMem().

The documentation for this class was generated from the following files:

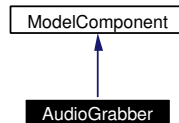
- [AttentionGuidanceMap.H](#)
- [AttentionGuidanceMap.C](#)

## 10.9 AudioGrabber Class Reference

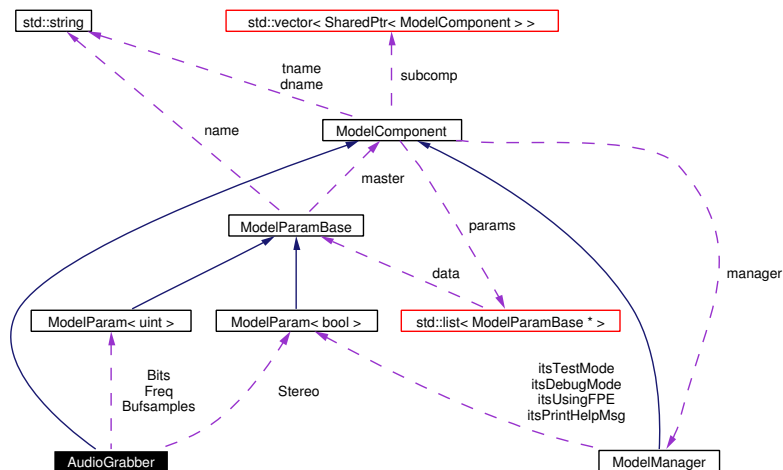
Class to grab audio samples from an audio device.

```
#include <AudioGrabber.H>
```

Inheritance diagram for AudioGrabber:



Collaboration diagram for AudioGrabber:



### Public Member Functions

- **AudioGrabber** (**ModelManager** &mgr, const **std::string** &descrName="Audio Grabber Driver", const **std::string** &tagName="AudioGrabber")

*Constructor.*

- **~AudioGrabber** ()

*Destructor.*

- virtual void **exportOptions** (const int whatToExport, const bool recurse=true)

Export some of our [ModelParam](#) parameters as command-line-options.

- int [grab](#) (const [byte](#) \*\*buffer) const

*Grab a buffer.*

## Protected Member Functions

- void [start2](#) ()

*get started (see [ModelComponent.H](#))*

- void [stop1](#) ()

*get stopped (see [ModelComponent.H](#))*

## Protected Attributes

- [ModelParam](#)< [std::string](#) > [DevName](#)

*audio device name*

- [ModelParam](#)< [uint](#) > [Bits](#)

*bits per sample (8 or 16)*

- [ModelParam](#)< [uint](#) > [Freq](#)

*sampling frequency (Hz)*

- [ModelParam](#)< [uint](#) > [Bufsamples](#)

*samples per buffer*

- [ModelParam](#)< [bool](#) > [Stereo](#)

*stereo recording?*

### 10.9.1 Detailed Description

Class to grab audio samples from an audio device.

For now, grabbing is done using a blocking read() onto the DSP device.

Definition at line 46 of file AudioGrabber.H.

## 10.9.2 Constructor & Destructor Documentation

### 10.9.2.1 `AudioGrabber::AudioGrabber (ModelManager & mgr, const std::string & descrName = "Audio Grabber Driver", const std::string & tagName = "AudioGrabber")`

Constructor.

Definition at line 50 of file AudioGrabber.C.

### 10.9.2.2 `AudioGrabber::~~AudioGrabber ()`

Destructor.

Definition at line 146 of file AudioGrabber.C.

## 10.9.3 Member Function Documentation

### 10.9.3.1 `void AudioGrabber::exportOptions (const int whatToExport, const bool recurse = true) [virtual]`

Export some of our [ModelParam](#) parameters as command-line-options.

See the base function in [ModelComponent.H](#) for info.

Reimplemented from [ModelComponent](#).

Definition at line 62 of file AudioGrabber.C.

References [Bits](#), [Bufsamples](#), [DevName](#), [ModelComponent::exportOptions\(\)](#), [Freq](#), [OPTEXP\\_CORE](#), [ModelManager::requestOption\(\)](#), and [Stereo](#).

### 10.9.3.2 `int AudioGrabber::grab (const byte ** buffer) const`

Grab a buffer.

returns buffer size, which is `Bufsamples * Bits/8 (*2 if Stereo)`

Definition at line 150 of file AudioGrabber.C.

References [ASSERT](#).

### 10.9.3.3 `void AudioGrabber::start2 () [protected, virtual]`

get started (see [ModelComponent.H](#))

Reimplemented from [ModelComponent](#).

Definition at line 78 of file AudioGrabber.C.

References [Bits](#), [Bufsamples](#), [DevName](#), [Freq](#), [ModelParam< bool >::getVal\(\)](#), [ModelParam< uint >::getVal\(\)](#), [ModelComponent::setModelParamVal\(\)](#), and [Stereo](#).

#### 10.9.3.4 void AudioGrabber::stop1 () [protected, virtual]

get stopped (see [ModelComponent.H](#))

Reimplemented from [ModelComponent](#).

Definition at line 138 of file AudioGrabber.C.

### 10.9.4 Member Data Documentation

#### 10.9.4.1 [ModelParam<uint>](#) AudioGrabber::Bits [protected]

bits per sample (8 or 16)

Definition at line 68 of file AudioGrabber.H.

Referenced by [exportOptions\(\)](#), and [start2\(\)](#).

#### 10.9.4.2 [ModelParam<uint>](#) AudioGrabber::Bufsamples [protected]

samples per buffer

Definition at line 70 of file AudioGrabber.H.

Referenced by [exportOptions\(\)](#), and [start2\(\)](#).

#### 10.9.4.3 [ModelParam<std::string>](#) AudioGrabber::DevName [protected]

audio device name

Definition at line 67 of file AudioGrabber.H.

Referenced by [exportOptions\(\)](#), and [start2\(\)](#).

#### 10.9.4.4 [ModelParam<uint>](#) AudioGrabber::Freq [protected]

sampling frequency (Hz)

Definition at line 69 of file AudioGrabber.H.

Referenced by [exportOptions\(\)](#), and [start2\(\)](#).

**10.9.4.5**   [ModelParam](#)<bool> [AudioGrabber::Stereo](#)   [protected]

stereo recording?

Definition at line 71 of file AudioGrabber.H.

Referenced by `exportOptions()`, and `start2()`.

The documentation for this class was generated from the following files:

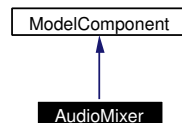
- [AudioGrabber.H](#)
- [AudioGrabber.C](#)

## 10.10 AudioMixer Class Reference

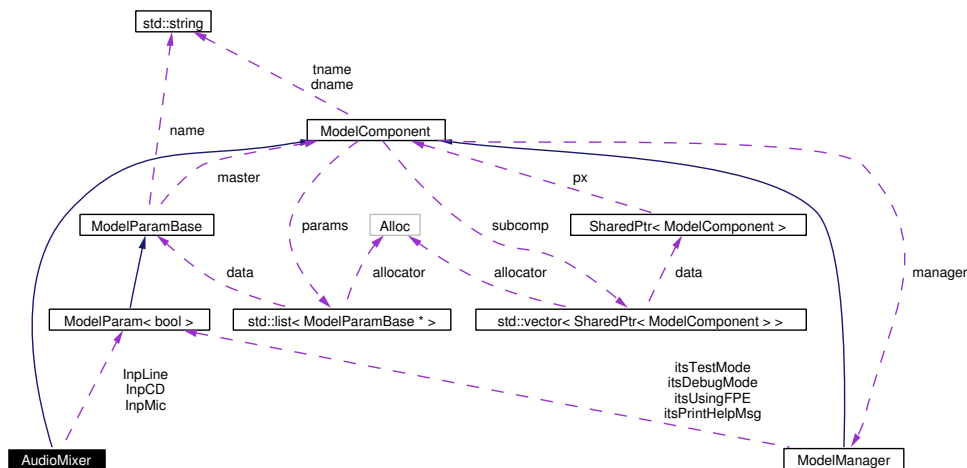
Simple interface to the Linux audio mixer.

```
#include <AudioMixer.H>
```

Inheritance diagram for AudioMixer:



Collaboration diagram for AudioMixer:



### Public Member Functions

- `AudioMixer` (`ModelManager` &mgr, const `std::string` &descrName="Audio Mixer Driver", const `std::string` &tagName="AudioMixer")

*Constructor.*

- `~AudioMixer` ()

*Destructor.*

- virtual void `exportOptions` (const int whatToExport, const bool recurse=true)

Export some of our [ModelParam](#) parameters as command-line-options.

## Protected Member Functions

- void [start2](#) ()  
*get started (see [ModelComponent.H](#))*
- void [stop1](#) ()  
*get stopped (see [ModelComponent.H](#))*

## Protected Attributes

- [ModelParam](#)< std::string > [DevName](#)  
*mixer device name*
- [ModelParam](#)< bool > [InpLine](#)  
*line input on*
- [ModelParam](#)< bool > [InpCD](#)  
*cd input on*
- [ModelParam](#)< bool > [InpMic](#)  
*mic input on*

### 10.10.1 Detailed Description

Simple interface to the Linux audio mixer.

To be used to set the recording input.

Definition at line 44 of file [AudioMixer.H](#).

### 10.10.2 Constructor & Destructor Documentation

#### 10.10.2.1 [AudioMixer::AudioMixer](#) ([ModelManager](#) & *mgr*, const std::string & *descrName* = "Audio Mixer Driver", const std::string & *tagName* = "AudioMixer")

Constructor.

Definition at line 50 of file [AudioMixer.C](#).



### 10.10.2.2 AudioMixer::~~AudioMixer ()

Destructor.

Definition at line 103 of file AudioMixer.C.

## 10.10.3 Member Function Documentation

### 10.10.3.1 void AudioMixer::exportOptions (const int *whatToExport*, const bool *recurse* = true) [virtual]

Export some of our [ModelParam](#) parameters as command-line-options.

See the base function in [ModelComponent.H](#) for info.

Reimplemented from [ModelComponent](#).

Definition at line 61 of file AudioMixer.C.

References [ModelComponent::exportOptions\(\)](#), [InpCD](#), [InpLine](#), [InpMic](#), [OPTEXP\\_CORE](#), and [ModelManager::requestOption\(\)](#).

### 10.10.3.2 void AudioMixer::start2 () [protected, virtual]

get started (see [ModelComponent.H](#))

Reimplemented from [ModelComponent](#).

Definition at line 76 of file AudioMixer.C.

References [ModelParam< bool >::getVal\(\)](#), [InpCD](#), [InpLine](#), and [InpMic](#).

### 10.10.3.3 void AudioMixer::stop1 () [protected, virtual]

get stopped (see [ModelComponent.H](#))

Reimplemented from [ModelComponent](#).

Definition at line 97 of file AudioMixer.C.

## 10.10.4 Member Data Documentation

### 10.10.4.1 [ModelParam<std::string>](#) AudioMixer::DevName [protected]

mixer device name

Definition at line 60 of file AudioMixer.H.

**10.10.4.2**   [ModelParam<bool> AudioMixer::InpCD](#)   [protected]

cd input on

Definition at line 62 of file AudioMixer.H.

Referenced by exportOptions(), and start2().

**10.10.4.3**   [ModelParam<bool> AudioMixer::InpLine](#)   [protected]

line input on

Definition at line 61 of file AudioMixer.H.

Referenced by exportOptions(), and start2().

**10.10.4.4**   [ModelParam<bool> AudioMixer::InpMic](#)   [protected]

mic input on

Definition at line 63 of file AudioMixer.H.

Referenced by exportOptions(), and start2().

The documentation for this class was generated from the following files:

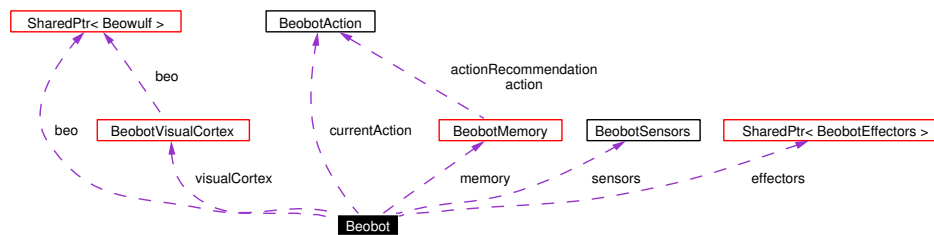
- [AudioMixer.H](#)
- [AudioMixer.C](#)

## 10.11 Beobot Class Reference

this class is a complete Beobot

```
#include <Beobot.H>
```

Collaboration diagram for Beobot:



### Public Member Functions

- **Beobot** (const char \*slaves, const int imgw, const int imgh, const int lev\_min, const int lev\_max, const int delta\_min, const int delta\_max, const int smlev, const int nborient, const **MaxNormType** normtype, const int jlev, const int jdepth, const int nbneig, const in\_addr\_t ip, const short int port)

*Constructor.*

- void **newVisualInput** (Image< **PixRGB**< byte > > &scene)  
*set new scene from an existing image*
- Image< **PixRGB**< byte > > \* **getRetinaPtr** ()  
*get pointer to internal retina, for example to directly framegrab it*
- void **lowLevel** (const int frame)  
*low level pyramid business*
- void **lowLevelStart** (const int frame)  
*start low-level by sending off current frame*
- void **lowLevelEnd** (const int frame)  
*receive results from slave for given frame*
- void **getWinner** (**Point2D** &win) const  
*get most salient location:*

- void [intermediateLevel](#) (bool takeYourTime=false)  
*compute the new "sensation" and update "memory"*
- void [highLevel](#) (void)  
*compute the new "action" and update "memory"*
- void [decision](#) (void)  
*take a decision*
- void [action](#) (void)  
*just do it*
- void [DEBUGgetClustered](#) (Image< [PixRGB](#)< [byte](#) > > &im)  
*returns the current clustered layout*
- void [DEBUGgetCurrentAction](#) ([BeobotAction](#) &a)  
*returns the current action*
- void [DEBUGsetCurrentAction](#) ([BeobotAction](#) &a)  
*set the current action*

### 10.11.1 Detailed Description

this class is a complete Beobot

i.e. Visual Cortex, Memory, Sensors and Effectors

Definition at line 47 of file Beobot.H.

### 10.11.2 Constructor & Destructor Documentation

**10.11.2.1 Beobot::Beobot (const char \* *slaves*, const int *imgw*, const int *imgh*, const int *lev\_min*, const int *lev\_max*, const int *delta\_min*, const int *delta\_max*, const int *smlev*, const int *nborient*, const [MaxNormType](#) *normtype*, const int *jlev*, const int *jdepth*, const int *nbneig*, const [in\\_addr\\_t](#) *ip*, const short int *port*)**

Constructor.

use NULL for slaves to use the single-CPU version

Definition at line 38 of file Beobot.C.

References [BeobotVisualCortex::init\(\)](#).

## 10.11.3 Member Function Documentation

### 10.11.3.1 void Beobot::action (void)

just do it

Definition at line 205 of file Beobot.C.

### 10.11.3.2 void Beobot::DEBUGgetClustered (Image< PixRGB< byte > > & im)

returns the current clustered layout

Definition at line 144 of file Beobot.C.

References BeobotSensation::getQuickLayout(), and BeobotMemory::passedSensation().

### 10.11.3.3 void Beobot::DEBUGgetCurrentAction (BeobotAction & a) [inline]

returns the current action

Definition at line 104 of file Beobot.H.

### 10.11.3.4 void Beobot::DEBUGsetCurrentAction (BeobotAction & a) [inline]

set the current action

Definition at line 108 of file Beobot.H.

### 10.11.3.5 void Beobot::decision (void)

take a decision

based of the recommendation issued by [highLevel\(\)](#) and passed sensations and actions

Definition at line 199 of file Beobot.C.

References BeobotMemory::generatePresentAction().

### 10.11.3.6 Image< PixRGB< byte > > \* Beobot::getRetinaPtr ()

get pointer to internal retina, for example to directly framegrab it

Definition at line 67 of file Beobot.C.

References `BeobotVisualCortex::getScenePtr()`.

Referenced by `main()`.

#### 10.11.3.7 `void Beobot::getWinner (Point2D & win) const`

get most salient location:

Definition at line 83 of file `Beobot.C`.

References `BeobotVisualCortex::getWinner()`.

#### 10.11.3.8 `void Beobot::highLevel (void)`

compute the new "action" and update "memory"

– higher level reasoning

Definition at line 153 of file `Beobot.C`.

References `centroid()`, `BeobotSensation::getCentroid()`, `BeobotSensation::getQuickLayout()`, `Image< T >::getWidth()`, `Point2D::i`, `Point2D::j`, `BeobotMemory::passedSensation()`, `BeobotMemory::presentActionRecommendation()`, `BeobotAction::setGear()`, `BeobotAction::setSpeed()`, and `BeobotAction::setTurn()`.

#### 10.11.3.9 `void Beobot::intermediateLevel (bool takeYourTime = false)`

compute the new "sensation" and update "memory"

( given a new scene i.e. after `see()` and someday the state of the sensors ) – high level vision but still low level cognitive processing `takeYourTime` is set to 'true' for the first frame which is necessary for convergence - see implementation

Definition at line 87 of file `Beobot.C`.

References `BeobotSensation::getCentroid()`, `BeobotVisualCortex::getClusteredImage()`, `BeobotVisualCortex::getInputSize()`, `Point2D::i`, `BeobotVisualCortex::initSprings()`, `BeobotVisualCortex::iterateSprings()`, `Point2D::j`, `BeobotMemory::passedSensation()`, and `BeobotMemory::presentSensation()`.

#### 10.11.3.10 `void Beobot::lowLevel (const int frame)`

low level pyramid business

When using parallel processing, this method will send off the current frame and wait until all results have come back. For better performance, use `lowLevelStart()` and `lowLevelEnd()` instead.

Definition at line 71 of file `Beobot.C`.

References BeobotVisualCortex::process().

Referenced by main().

#### 10.11.3.11 void Beobot::lowLevelEnd (const int *frame*)

receive results from slave for given frame

Definition at line 79 of file Beobot.C.

References BeobotVisualCortex::processEnd().

Referenced by main().

#### 10.11.3.12 void Beobot::lowLevelStart (const int *frame*)

start low-level by sending off current frame

Definition at line 75 of file Beobot.C.

References BeobotVisualCortex::processStart().

Referenced by main().

#### 10.11.3.13 void Beobot::newVisualInput ([Image](#)< [PixRGB](#)< [byte](#) > > & *scene*)

set new scene from an existing image

Definition at line 63 of file Beobot.C.

References BeobotVisualCortex::newVisualInput().

The documentation for this class was generated from the following files:

- [Beobot.H](#)
- [Beobot.C](#)

## 10.12 BeobotAction Class Reference

This class defined motor actions for the Beobots.

```
#include <BeobotAction.H>
```

### Public Member Functions

- [BeobotAction](#) ()  
*no arg constructor*
- [BeobotAction](#) (const float t, const float s, const int g)  
*constructor*
- [BeobotAction](#) (const [BeobotAction](#) &other)  
*copy constructor*
- [BeobotAction](#) & [operator=](#) (const [BeobotAction](#) &other)  
*Assignment constructor.*
- float [getTurn](#) () const  
*access function*
- float [getSpeed](#) () const  
*access function*
- int [getGear](#) () const  
*access function*
- void [setTurn](#) (const float t)  
*access function*
- void [setSpeed](#) (const float s)  
*access function*
- void [setGear](#) (const int g)  
*access function*
- [~BeobotAction](#) ()  
*destructor*



### 10.12.1 Detailed Description

This class defined motor actions for the Beobots.

Turn is between -1.0 (full left) and 1.0 (full right), speed is between -1.0 (full reverse) and 1.0 (full forward), and gear is 0 (neutral), 1 or 2.

Definition at line 48 of file BeobotAction.H.

### 10.12.2 Constructor & Destructor Documentation

#### 10.12.2.1 BeobotAction::BeobotAction () [inline]

no arg constructor

Definition at line 98 of file BeobotAction.H.

#### 10.12.2.2 BeobotAction::BeobotAction (const float *t*, const float *s*, const int *g*) [inline]

constructor

Definition at line 106 of file BeobotAction.H.

References `setGear()`, `setSpeed()`, and `setTurn()`.

#### 10.12.2.3 BeobotAction::BeobotAction (const BeobotAction & *other*) [inline]

copy constructor

Definition at line 102 of file BeobotAction.H.

References `gear`, `speed`, and `turn`.

#### 10.12.2.4 BeobotAction::~~BeobotAction () [inline]

destructor

Definition at line 114 of file BeobotAction.H.

### 10.12.3 Member Function Documentation

#### 10.12.3.1 int BeobotAction::getGear () const [inline]

access function

Definition at line 126 of file BeobotAction.H.

Referenced by BeobotEffectors::performAction().

#### 10.12.3.2 float BeobotAction::getSpeed () const [inline]

access function

Definition at line 122 of file BeobotAction.H.

Referenced by BeobotMemory::generatePresentAction(), main(), and BeobotEffectors::performAction().

#### 10.12.3.3 float BeobotAction::getTurn () const [inline]

access function

Definition at line 118 of file BeobotAction.H.

Referenced by BeobotMemory::generatePresentAction(), main(), and BeobotEffectors::performAction().

#### 10.12.3.4 BeobotAction & BeobotAction::operator= (const BeobotAction & other) [inline]

Assignment constructor.

Definition at line 110 of file BeobotAction.H.

References gear, speed, and turn.

#### 10.12.3.5 void BeobotAction::setGear (const int g) [inline]

access function

Definition at line 144 of file BeobotAction.H.

Referenced by BeobotAction(), BeobotMemory::generatePresentAction(), Beobot::highLevel(), and main().

#### 10.12.3.6 void BeobotAction::setSpeed (const float s) [inline]

access function

Definition at line 137 of file BeobotAction.H.

Referenced by BeobotAction(), BeobotMemory::generatePresentAction(), Beobot::highLevel(), and main().

**10.12.3.7 void BeobotAction::setTurn (const float t) [inline]**

access function

Definition at line 130 of file BeobotAction.H.

Referenced by BeobotAction(), BeobotMemory::generatePresentAction(), Beobot::highLevel(), and main().

The documentation for this class was generated from the following file:

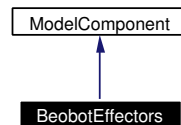
- [BeobotAction.H](#)

## 10.13 BeobotEffectors Class Reference

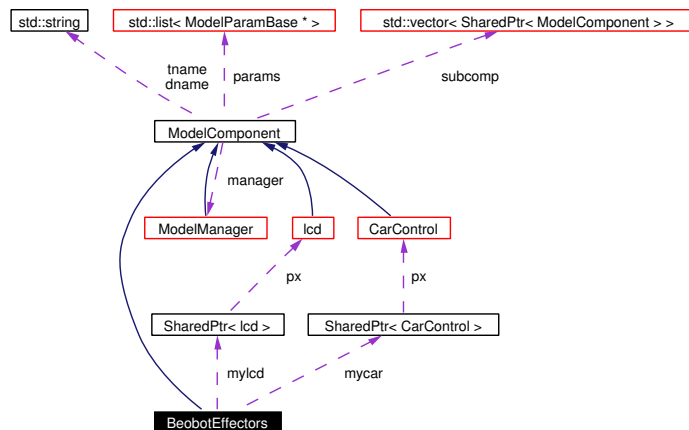
This class contains all the effectors of a [Beobot](#).

```
#include <BeobotEffectors.H>
```

Inheritance diagram for BeobotEffectors:



Collaboration diagram for BeobotEffectors:



### Public Member Functions

- [BeobotEffectors](#) ([ModelManager](#) &mgr, const [std::string](#) &descrName="Beobot Effectors", const [std::string](#) &tagName="BeobotEffectors")  
*constructor*
- [~BeobotEffectors](#) ()  
*destructor*
- void [performAction](#) ([BeobotAction](#) &action)  
*execute a given action*

## Protected Attributes

- `SharedPtr< CarControl > mycar`  
*our car*
- `SharedPtr< lcd > mylcd`  
*our LCD screen*

### 10.13.1 Detailed Description

This class contains all the effectors of a [Beobot](#).  
i.e. servos (speed, turn and gear) and LCD display  
Definition at line 47 of file BeobotEffectors.H.

### 10.13.2 Constructor & Destructor Documentation

**10.13.2.1** `BeobotEffectors::BeobotEffectors (ModelManager & mgr, const std::string & descrName = "Beobot Effectors", const std::string & tagName = "BeobotEffectors")`

constructor

Definition at line 44 of file BeobotEffectors.C.

References `ModelComponent::addSubComponent()`, `mycar`, and `mylcd`.

**10.13.2.2** `BeobotEffectors::~~BeobotEffectors ()`

destructor

Definition at line 57 of file BeobotEffectors.C.

### 10.13.3 Member Function Documentation

**10.13.3.1** `void BeobotEffectors::performAction (BeobotAction & action)`

execute a given action

Definition at line 61 of file BeobotEffectors.C.

References `BeobotAction::getGear()`, `BeobotAction::getSpeed()`, and `BeobotAction::getTurn()`.

## 10.13.4 Member Data Documentation

### 10.13.4.1 [SharedPtr<CarControl>](#) [BeobotEffectors::mycar](#) [protected]

our car

Definition at line 61 of file BeobotEffectors.H.

Referenced by BeobotEffectors().

### 10.13.4.2 [SharedPtr<lcd>](#) [BeobotEffectors::mylcd](#) [protected]

our LCD screen

Definition at line 62 of file BeobotEffectors.H.

Referenced by BeobotEffectors().

The documentation for this class was generated from the following files:

- [BeobotEffectors.H](#)
- [BeobotEffectors.C](#)

## 10.14 BeobotMemory Class Reference

Memory of a [Beobot](#).

```
#include <BeobotMemory.H>
```

Collaboration diagram for BeobotMemory:

### Public Member Functions

- void [presentSensation](#) ([BeobotSensation](#) &sensa)  
*set present sensation*
- void [presentActionRecommendation](#) ([BeobotAction](#) &act)  
*set present action*
- bool [passedSensation](#) (int howOld, [BeobotSensation](#) &sensa)  
*get sensation at time t-howOld - false iif nothing to remember*
- void [generatePresentAction](#) ([BeobotAction](#) &currentAction)  
*generate action based on recommendation and previous actions*

### Static Public Attributes

- const int **memoryLength** = 6

#### 10.14.1 Detailed Description

Memory of a [Beobot](#).

this is more or less a FIFO queue of actions & sensation actions are computed from the recommended actions proposed by [Beobot::highLevel\(\)](#)

Definition at line 52 of file BeobotMemory.H.

#### 10.14.2 Member Function Documentation

##### 10.14.2.1 void BeobotMemory::generatePresentAction ([BeobotAction](#) &currentAction)

generate action based on recommendation and previous actions

Definition at line 92 of file BeobotMemory.C.

References ASSERT, BeobotAction::getSpeed(), BeobotAction::getTurn(), BeobotAction::setGear(), BeobotAction::setSpeed(), and BeobotAction::setTurn().

Referenced by Beobot::decision().

#### 10.14.2.2 **bool BeobotMemory::passedSensation (int *howOld*, [BeobotSensation](#) & *sensa*)**

get sensation at time t-howOld - false iif nothing to remember

Definition at line 78 of file BeobotMemory.C.

References ASSERT.

Referenced by Beobot::DEBUGgetClustered(), Beobot::highLevel(), and Beobot::intermediateLevel().

#### 10.14.2.3 **void BeobotMemory::presentActionRecommendation ([BeobotAction](#) & *act*)**

set present action

Definition at line 67 of file BeobotMemory.C.

References ASSERT.

Referenced by Beobot::highLevel().

#### 10.14.2.4 **void BeobotMemory::presentSensation ([BeobotSensation](#) & *sensa*)**

set present sensation

Definition at line 53 of file BeobotMemory.C.

References ASSERT.

Referenced by Beobot::intermediateLevel().

The documentation for this class was generated from the following files:

- [BeobotMemory.H](#)
- [BeobotMemory.C](#)

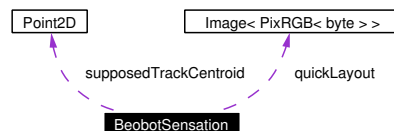


## 10.15 BeobotSensation Class Reference

this class contains all the sensory informations but not the action

```
#include <BeobotSensation.H>
```

Collaboration diagram for BeobotSensation:



### Public Member Functions

- [BeobotSensation](#) ()  
*no arg constructor*
- [BeobotSensation](#) (const [Image< PixRGB< byte > >](#) &quicklayout, const [Point2D](#) &supposedcrackcentroid)  
*constructor*
- [BeobotSensation](#) (const [BeobotSensation](#) &other)  
*copy constructor*
- [BeobotSensation](#) & [operator=](#) (const [BeobotSensation](#) &other)  
*assignment constructor:*
- void [getQuickLayout](#) ([Image< PixRGB< byte > >](#) &im) const  
*access function*
- void [getCentroid](#) ([Point2D](#) &centroid) const  
*access function*
- void [setConfused](#) (void)  
*set 'confused' to 'true'*
- bool [isConfused](#) (void)  
*return 'confused'*
- bool [isInitialized](#) (void)  
*return 'initialized'*

- [~BeobotSensation \(\)](#)

*destructor*

### 10.15.1 Detailed Description

this class contains all the sensory informations but not the action

it is used to compute an action

Definition at line 47 of file BeobotSensation.H.

### 10.15.2 Constructor & Destructor Documentation

#### 10.15.2.1 BeobotSensation::BeobotSensation () [inline]

no arg constructor

Definition at line 99 of file BeobotSensation.H.

#### 10.15.2.2 BeobotSensation::BeobotSensation (const [Image](#)< [PixRGB](#)< [byte](#) > > & *quicklayout*, const [Point2D](#) & *supposedcrackcentroid*) [inline]

constructor

Definition at line 103 of file BeobotSensation.H.

References [ASSERT](#).

#### 10.15.2.3 BeobotSensation::BeobotSensation (const [BeobotSensation](#) & *other*)

copy constructor

Definition at line 46 of file BeobotSensation.C.

References [ASSERT](#), [confused](#), [initialized](#), [quickLayout](#), and [supposedTrackCentroid](#).

#### 10.15.2.4 BeobotSensation::~BeobotSensation ()

destructor

Definition at line 40 of file BeobotSensation.C.

References [Image](#)< [PixRGB](#)< [byte](#) > >::freeMem().

### 10.15.3 Member Function Documentation

#### 10.15.3.1 void BeobotSensation::getCentroid ([Point2D](#) & *centroid*) const [inline]

access function

Definition at line 125 of file BeobotSensation.H.

References ASSERT, and centroid().

Referenced by Beobot::highLevel(), and Beobot::intermediateLevel().

#### 10.15.3.2 void BeobotSensation::getQuickLayout ([Image](#)< [PixRGB](#)< [byte](#) > > & *im*) const [inline]

access function

Definition at line 118 of file BeobotSensation.H.

References ASSERT.

Referenced by Beobot::DEBUGgetClustered(), and Beobot::highLevel().

#### 10.15.3.3 bool BeobotSensation::isConfused (void) [inline]

return 'confused'

Definition at line 73 of file BeobotSensation.H.

#### 10.15.3.4 bool BeobotSensation::isInitialized (void) [inline]

return 'initialized'

Definition at line 76 of file BeobotSensation.H.

#### 10.15.3.5 [BeobotSensation](#) & BeobotSensation::operator= (const [BeobotSensation](#) & *other*)

assignment constructor:

Definition at line 57 of file BeobotSensation.C.

References ASSERT, confused, initialized, quickLayout, and supposedTrackCentroid.

**10.15.3.6 void BeobotSensation::setConfused (void) [inline]**

set 'confused' to 'true'

Definition at line 70 of file BeobotSensation.H.

The documentation for this class was generated from the following files:

- [BeobotSensation.H](#)
- [BeobotSensation.C](#)

## 10.16 BeobotSensors Class Reference

All the sensors of a [Beobot](#).

```
#include <BeobotSensors.H>
```

### 10.16.1 Detailed Description

All the sensors of a [Beobot](#).

speed sensor and rotation maybe a keyboard someday

Definition at line 43 of file BeobotSensors.H.

The documentation for this class was generated from the following file:

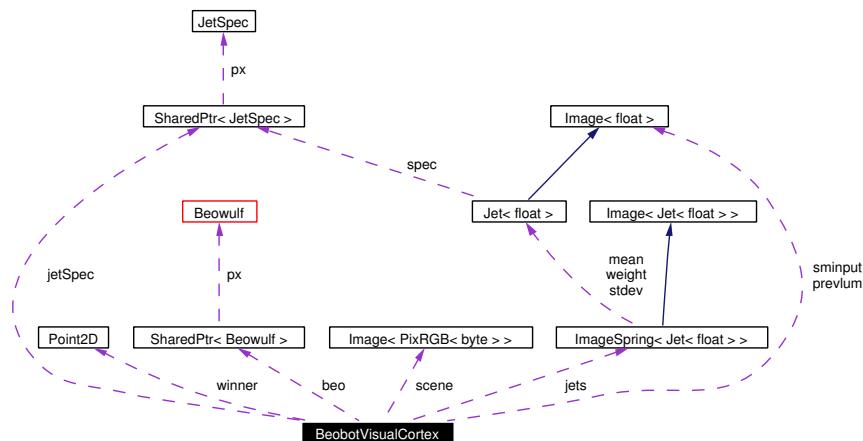
- [BeobotSensors.H](#)

## 10.17 BeobotVisualCortex Class Reference

Visual Cortex of a [Beobot](#).

```
#include <BeobotVisualCortex.H>
```

Collaboration diagram for BeobotVisualCortex:



### Public Member Functions

- [BeobotVisualCortex](#) ()  
*constructor*
- void [init](#) (const int imgw, const int imgh, const int lev\_min, const int lev\_max, const int delta\_min, const int delta\_max, const int smlev, const int nborient, const [MaxNormType](#) normtype, const int jlev, const int jdepth, const int nbneig, [SharedPtr](#)< [Beowulf](#) > beow)  
*initialization*
- void [newVisualInput](#) ([Image](#)< [PixRGB](#)< [byte](#) > > &scene)  
*present a new visual input from an existing image*
- [Image](#)< [PixRGB](#)< [byte](#) > > \* [getScenePtr](#) ()  
*get pointer to scene image, so that we can directly grab it*
- void [process](#) (const int frame)  
*call this to execute all the low-level processing on current scene*

- void [processStart](#) (const int frame)  
*start parallel processing by sending off current frame*
- void [processEnd](#) (const int frame)  
*end parallel process by receiving results*
- void [singleCPUprocess](#) (const int frame)  
*not normally used; use [process\(\)](#) instead*
- void [masterProcess](#) (const int frame)  
*not normally used; use [process\(\)](#) instead*
- void [slaveProcess](#) ()  
*Call this in an infinite loop on slaves to process & return incoming maps.*
- void [masterCollect](#) ()  
*not normally used; use [process\(\)](#) instead*
- void [getWinner](#) ([Point2D](#) &win) const  
*get most salient location:*
- void [initSprings](#) (bool initPosMasses)  
*initialize clustering with "spring method"*
- void [iterateSprings](#) (const float dt)  
*iterate clustering with "spring method"*
- void [getClusteredImage](#) ([Image](#)< [PixRGB](#)< [byte](#) > > &clusteredImage, [Point2D](#) &supposedTrackCentroid, const [Point2D](#) &previousTrackCentroid)  
*returns the clustered image and the position of the track centroid*
- void [getInputSize](#) ([Point2D](#) &size)  
*get size of input image*
- void [getPositions](#) ([Image](#)< [PixRGB](#)< [byte](#) > > &img, const int zoom)  
*get mass positions*

### 10.17.1 Detailed Description

Visual Cortex of a [Beobot](#).

this is no longer derived from [VisualCortex](#) it can be used with one CPU or SMP systems

Definition at line 53 of file BeobotVisualCortex.H.

## 10.17.2 Constructor & Destructor Documentation

### 10.17.2.1 BeobotVisualCortex::BeobotVisualCortex ()

constructor

Definition at line 50 of file BeobotVisualCortex.C.

## 10.17.3 Member Function Documentation

### 10.17.3.1 void BeobotVisualCortex::getClusteredImage (Image< [PixRGB](#)< [byte](#) > > & clusteredImage, [Point2D](#) & supposedTrackCentroid, const [Point2D](#) & previousTrackCentroid)

returns the clustered image and the position of the track centroid

Definition at line 440 of file BeobotVisualCortex.C.

References [ImageSpring](#)< [Jet](#)< float > >::getClusteredImage().

Referenced by [Beobot::intermediateLevel\(\)](#).

### 10.17.3.2 void BeobotVisualCortex::getInputSize ([Point2D](#) & size) [inline]

get size of input image

Definition at line 163 of file BeobotVisualCortex.H.

References [ASSERT](#), [Image](#)< [PixRGB](#)< byte > >::getHeight(), [Image](#)< [PixRGB](#)< byte > >::getWidth(), [Point2D](#)::i, [Image](#)< [PixRGB](#)< byte > >::initialized(), and [Point2D](#)::j.

Referenced by [Beobot::intermediateLevel\(\)](#).

### 10.17.3.3 void BeobotVisualCortex::getPositions (Image< [PixRGB](#)< [byte](#) > > & img, const int zoom)

get mass positions

Definition at line 451 of file BeobotVisualCortex.C.

References [ImageSpring](#)< [Jet](#)< float > >::getPositions().



**10.17.3.4 Image< PixRGB< byte > > \* BeobotVisualCortex::getScenePtr ()**

get pointer to scene image, so that we can directly grab it

Definition at line 91 of file BeobotVisualCortex.C.

Referenced by Beobot::getRetinaPtr().

**10.17.3.5 void BeobotVisualCortex::getWinner (Point2D & win) const**

get most salient location:

Definition at line 427 of file BeobotVisualCortex.C.

References Point2D::i, and Point2D::j.

Referenced by Beobot::getWinner().

**10.17.3.6 void BeobotVisualCortex::init (const int imgw, const int imgh, const int lev\_min, const int lev\_max, const int delta\_min, const int delta\_max, const int smlev, const int nborient, const MaxNormType normtype, const int jlev, const int jdepth, const int nbneig, SharedPtr< Beowulf > beow)**

initialization

use beow=NULL for single-CPU processing

Definition at line 54 of file BeobotVisualCortex.C.

References JetSpec::addIndexRange(), Image< Jet< float > >::beginw(), BY, Image< Jet< float > >::endw(), ImageSpring< T >::init(), ImageSpring< Jet< float > >::init(), INTENS, makeShared(), ORI, JetSpec::print(), RAW, Image< float >::resize(), Image< PixRGB< byte > >::resize(), and RG.

Referenced by Beobot::Beobot().

**10.17.3.7 void BeobotVisualCortex::initSprings (bool initPosMasses)**

initialize clustering with "spring method"

Definition at line 431 of file BeobotVisualCortex.C.

References ImageSpring< Jet< float > >::initClustering().

Referenced by Beobot::intermediateLevel().

**10.17.3.8 void BeobotVisualCortex::iterateSprings (const float dt)**

iterate clustering with "spring method"

Definition at line 436 of file BeobotVisualCortex.C.

References ImageSpring< Jet< float > >::computePos().

Referenced by Beobot::intermediateLevel().

**10.17.3.9 void BeobotVisualCortex::masterCollect ()**

not normally used; use [process\(\)](#) instead

Definition at line 329 of file BeobotVisualCortex.C.

References ASSERT, Image< Jet< float > >::beginw(), BY, COLOR, FLICKER, TCPmessage::getElement(), Image< PixRGB< byte > >::getHeight(), Image< Jet< float > >::getHeight(), Image< PixRGB< byte > >::getWidth(), Image< Jet< float > >::getWidth(), INTENS, ORI, RAW, RG, and VisualFeature.

Referenced by process(), and processEnd().

**10.17.3.10 void BeobotVisualCortex::masterProcess (const int frame)**

not normally used; use [process\(\)](#) instead

Definition at line 218 of file BeobotVisualCortex.C.

References TCPmessage::addImage(), ASSERT, getRGBY(), lum, luminance(), TCPmessage::reset(), and TCPmessage::setAction().

Referenced by process(), and processStart().

**10.17.3.11 void BeobotVisualCortex::newVisualInput (Image< PixRGB< byte > > & scene)**

present a new visual input from an existing image

Definition at line 87 of file BeobotVisualCortex.C.

References Image< PixRGB< byte > >::getHeight(), and Image< PixRGB< byte > >::getWidth().

Referenced by Beobot::newVisualInput().

**10.17.3.12 void BeobotVisualCortex::process (const int frame)**

call this to execute all the low-level processing on current scene

This method will automatically parallelize the processing if a non-NULL beowulf was passed at initialization. However, it will call [processStart\(\)](#) and [processEnd\(\)](#) in sequence, waiting for the results to be ready before returning. It may be a better idea to separately call [processStart\(\)](#), do something, then [processEnd\(\)](#). For single-CPU operation, [process\(\)](#) is the only option.

Definition at line 95 of file BeobotVisualCortex.C.

References [Image< float >::clear\(\)](#), [Image< float >::findMax\(\)](#), [SharedPtr< Beowulf >::get\(\)](#), [Point2D::i](#), [Point2D::j](#), [masterCollect\(\)](#), [masterProcess\(\)](#), [maxNormalize\(\)](#), and [singleCPUprocess\(\)](#).

Referenced by [Beobot::lowLevel\(\)](#).

#### 10.17.3.13 void BeobotVisualCortex::processEnd (const int *frame*)

end parallel process by receiving results

Definition at line 130 of file BeobotVisualCortex.C.

References [Image< float >::clear\(\)](#), [Image< float >::findMax\(\)](#), [Timer::get\(\)](#), [SharedPtr< Beowulf >::get\(\)](#), [Point2D::i](#), [Point2D::j](#), [masterCollect\(\)](#), and [maxNormalize\(\)](#).

Referenced by [Beobot::lowLevelEnd\(\)](#).

#### 10.17.3.14 void BeobotVisualCortex::processStart (const int *frame*)

start parallel processing by sending off current frame

Definition at line 122 of file BeobotVisualCortex.C.

References [SharedPtr< Beowulf >::get\(\)](#), and [masterProcess\(\)](#).

Referenced by [Beobot::lowLevelStart\(\)](#).

#### 10.17.3.15 void BeobotVisualCortex::singleCPUprocess (const int *frame*)

not normally used; use [process\(\)](#) instead

Definition at line 153 of file BeobotVisualCortex.C.

References [Image< Jet< float > >::beginw\(\)](#), [buildPyrGaussian\(\)](#), [buildPyrOriented\(\)](#), [BY](#), [byte](#), [Image< Jet< float > >::getHeight\(\)](#), [ImageSet< T >::getPixel\(\)](#), [getRGBY\(\)](#), [Image< Jet< float > >::getWidth\(\)](#), [Point2D::i](#), [Image< float >::initialized\(\)](#), [INTENS](#), [Point2D::j](#), [luminance\(\)](#), [ORI](#), [RAW](#), [Image< float >::resize\(\)](#), [RG](#), and [Image< T >::setVal\(\)](#).

Referenced by [process\(\)](#).

**10.17.3.16 void BeobotVisualCortex::slaveProcess ()**

Call this in an infinite loop on slaves to process & return incoming maps.

Definition at line 246 of file BeobotVisualCortex.C.

References `ASSERT`, `Gaussian5`, `TCPmessage::getElement()`, `Image< float >::initialized()`, and `Oriented5`.

The documentation for this class was generated from the following files:

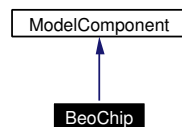
- [BeobotVisualCortex.H](#)
- [BeobotVisualCortex.C](#)

## 10.18 BeoChip Class Reference

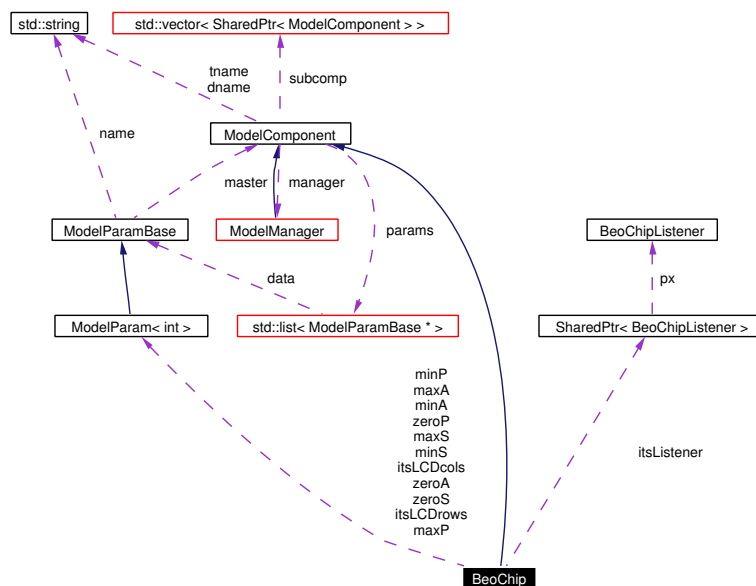
[BeoChip.H](#) Interface to Brian Hudson's BeoChip device.

```
#include <BeoChip.H>
```

Inheritance diagram for BeoChip:



Collaboration diagram for BeoChip:



### Public Member Functions

- void [run](#) ()  
*This is our main running thread - don't call directly.*
- bool [writeByte](#) (const [byte](#) val)  
*send a byte to the BeoChip*

### Constructors, destructors and debugging

- [BeoChip](#) ([ModelManager](#) &mgr, const [std::string](#) &descrName="Beo-Chip IO subsystem", const [std::string](#) &tagName="BeoChip", const char \*dev="/dev/ttyS0")  
*Default constructor. See [ModelComponent.H](#).*
- [~BeoChip](#) ()  
*Destructor.*
- void [setListener](#) ([SharedPtr](#)< [BeoChipListener](#) > &listener)  
*Install a callback (listener) for BeoChip events.*
- bool [echoRequest](#) ()  
*Send an echo request.*
- bool [debugMode](#) (const bool on)  
*Turn debug mode on/off.*
- bool [resetChip](#) ()  
*Reset the BeoChip.*

### LCD functions

- bool [lcdGotoXY](#) (const int x=0, const int y=0)  
*Go to given cursor position.*
- bool [lcdGoto](#) (const int i=0)  
*Go to given cursor position.*
- bool [lcdPrintf](#) (const char \*fmt,...)  
*Print some text.*
- bool [lcdPrintf](#) (const int x, const int y, const char \*fmt,...)  
*Print some text starting at given cursor position.*
- bool [lcdClear](#) ()  
*Clear LCD screen.*
- bool [lcdScrollLeft](#) (const int i=1)  
*Scroll display left by i positions.*
- bool [lcdScrollRight](#) (const int i=1)  
*Scroll display right by i positions.*

- bool `lcdMoveCursorLeft` (const int i=1)  
*Move cursor left by i positions.*
- bool `lcdMoveCursorRight` (const int i=1)  
*Move cursor right by i positions.*
- bool `lcdCursorBlock` ()  
*Make cursor a blinking block.*
- bool `lcdCursorUnderline` ()  
*Make cursor a blinking underline.*
- bool `lcdCursorInvisible` ()  
*Make cursor invisible (not recommended).*
- bool `lcdLoadFont` (const int font)  
*Load one of the pre-programmed custom fonts.*
- bool `lcdLoadFont` (const byte data[64])  
*Load a new font from an array of 64 bytes.*
- bool `lcdSetAnimation` (const int anim=0)  
*Select an LCD animation and start it.*
- bool `lcdSendRaw` (const byte val, const bool RS)  
*Send a raw byte to the LCD.*

#### Servo control functions

- bool `shimServos` (const byte shim)  
*Shim the BeoChip's servo pulses.*
- void `calibrateServo` (const int servo, const byte neutralval, const byte minval, const byte maxval)  
*Calibrate a servo.*
- bool `setServo` (const int servo, const float position)  
*Moves servo # to given position in [-1.0 .. 1.0].*
- float `getServo` (const int servo) const  
*Gets the current position of given servo.*
- bool `setServoRaw` (const int servo, const byte val)  
*Sets servo number servo to value val.*

- `byte getServoRaw` (const int servo) const  
*Gets the current raw position of given servo.*

#### Pulse acquisition functions

- `bool capturePulse` (const int whichone, const bool on)  
*Turn pulse captures on/off.*
- `void calibratePulse` (const int whichone, const int neutralval, const int minval, const int maxval)  
*Calibrate a pulse.*
- `float getPulse` (const int whichone)  
*Get current pulse value, between [-1.0..1.0].*
- `short int getPulseRaw` (const int whichone)  
*Get current pulse value, raw uncalibrated.*

#### Analog acquisition functions

- `bool captureAnalog` (const int whichone, const bool on)  
*Turn A/D captures on/off.*
- `void calibrateAnalog` (const int whichone, const int neutralval, const int minval, const int maxval)  
*Calibrate an A/D converter.*
- `float getAnalog` (const int whichone)  
*Get current calibrated Analog value.*
- `byte getAnalogRaw` (const int whichone)  
*Get current raw Analog value.*

#### Keyboard acquisition functions

- `bool captureKeyboard` (const bool on)  
*Turn keyboard capture on/off.*
- `bool debounceKeyboard` (const bool on)  
*Turn keyboard debouncing on/off.*
- `int getKeyboard` ()  
*Get current Keyboard value.*



### Digital output function

- bool [setDigitalOut](#) (const int outnum, const bool on)  
*Turn a given digital output on/off.*

### Protected Member Functions

- float [rawToCalib](#) (const int raw, const int zero, const int mini, const int maxi) const  
*Convert from raw (int) to calibrated (-1.0..1.0) position.*
- int [calibToRaw](#) (const float calibrated, const int zero, const int mini, const int maxi, const int bits=8) const  
*Convert from calibrated (-1.0..1.0) to raw (int) position.*
- void [start1](#) ()  
*open the port and get started*
- void [stop2](#) ()  
*close the port and get stopped*

### Protected Attributes

- [ModelParam](#)< std::string > [itsDevName](#)  
*name of our serial device*
- [ModelParam](#)< int > [itsLCDrows](#)  
*number of rows of LCD screen (lines of text)*
- [ModelParam](#)< int > [itsLCDcols](#)  
*number of columns of LCD screen*
- [ModelParam](#)< int > \* [zeroS](#)  
*zero calibration value, for 0.0F (servo)*
- [ModelParam](#)< int > \* [minS](#)  
*minimum raw value, for -1.0F (servo)*

- [ModelParam](#)< int > \* [maxS](#)  
*maximum raw value, for 1.0F (servo)*
- [byte](#) \* [servopos](#)  
*raw servo positions [0..255]*
- [ModelParam](#)< int > \* [zeroP](#)  
*zero calibration value, for 0.0F (pulse)*
- [ModelParam](#)< int > \* [minP](#)  
*minimum raw value, for -1.0F (pulse)*
- [ModelParam](#)< int > \* [maxP](#)  
*maximum raw value, for 1.0F (pulse)*
- [short int](#) \* [pulseval](#)  
*raw pulse values (11bit int)*
- [ModelParam](#)< int > \* [zeroA](#)  
*zero calibration value, for 0.0F (pulse)*
- [ModelParam](#)< int > \* [minA](#)  
*minimum raw value, for -1.0F (servo)*
- [ModelParam](#)< int > \* [maxA](#)  
*maximum raw value, for 1.0F (servo)*
- [byte](#) \* [adcval](#)  
*raw adc values [0..255]*
- [byte](#) [keyboard](#)  
*our current keyboard state*

### 10.18.1 Detailed Description

[BeoChip.H](#) Interface to Brian Hudson's BeoChip device.

Hardware interface Class. Contains interfaces to display to an LCD, read servo pulse values (2), set servo positions (8), get A/D conversion values (2), manipulate four digital outs, and finally, get some input from 5 pushbuttons.

Definition at line 100 of file BeoChip.H.

## 10.18.2 Constructor & Destructor Documentation

### 10.18.2.1 BeoChip::BeoChip ([ModelManager](#) & *mgr*, const [std::string](#) & *descrName* = "BeoChip IO subsystem", const [std::string](#) & *tagName* = "BeoChip", const char \* *dev* = "/dev/ttyS0")

Default constructor. See [ModelComponent.H](#).

Definition at line 74 of file BeoChip.C.

References [adcval](#), [ModelParam](#)< int >::init(), [keyboard](#), [maxA](#), [maxP](#), [maxS](#), [minA](#), [minP](#), [minS](#), [pulseval](#), [SharedPtr](#)< [BeoChipListener](#) >::reset(), [servopos](#), [uint](#), [zeroA](#), [zeroP](#), and [zeroS](#).

### 10.18.2.2 BeoChip::~BeoChip ()

Destructor.

Definition at line 173 of file BeoChip.C.

References [adcval](#), [maxA](#), [maxP](#), [maxS](#), [minA](#), [minP](#), [minS](#), [pulseval](#), [servopos](#), [zeroA](#), [zeroP](#), and [zeroS](#).

## 10.18.3 Member Function Documentation

### 10.18.3.1 void BeoChip::calibrateAnalog (const int *whichone*, const int *neutralval*, const int *minval*, const int *maxval*)

Calibrate an A/D converter.

Calibration will be made so that [getAnalog\(\)](#) returns 0.0F when raw value width is 'neutralval', returns -1.0F when the raw value is minval and returns 1.0F when the raw value is maxval.

Definition at line 517 of file BeoChip.C.

References [ASSERT](#), [maxA](#), [minA](#), [ModelParam](#)< int >::setVal(), and [zeroA](#).

### 10.18.3.2 void BeoChip::calibratePulse (const int *whichone*, const int *neutralval*, const int *minval*, const int *maxval*)

Calibrate a pulse.

Calibration will be made so that [getPulse\(\)](#) returns 0.0F when raw pulse width is 'neutralval', returns -1.0F when the raw pulse value is minval and returns 1.0F when the raw pulse value is maxval.

Definition at line 479 of file BeoChip.C.

References ASSERT, maxP, minP, ModelParam< int >::setVal(), and zeroP.

#### **10.18.3.3 void BeoChip::calibrateServo (const int *servo*, const byte *neutralval*, const byte *minval*, const byte *maxval*)**

Calibrate a servo.

Calibration will be made so that setServo(servo, 0.0F) will send the value 'neutralval' to the servo, setServo(servo, -1.0F) will send minval and setServo(servo, 1.0F) will send maxval.

Definition at line 418 of file BeoChip.C.

References ASSERT, maxS, minS, ModelParam< int >::setVal(), and zeroS.

#### **10.18.3.4 int BeoChip::calibToRaw (const float *calibrated*, const int *zero*, const int *mini*, const int *maxi*, const int *bits* = 8) const [protected]**

Convert from calibrated (-1.0..1.0) to raw (int) position.

Definition at line 710 of file BeoChip.C.

Referenced by setServo().

#### **10.18.3.5 bool BeoChip::captureAnalog (const int *whichone*, const bool *on*)**

Turn A/D captures on/off.

Definition at line 508 of file BeoChip.C.

References ASSERT, and writeByte().

#### **10.18.3.6 bool BeoChip::captureKeyboard (const bool *on*)**

Turn keyboard capture on/off.

Definition at line 546 of file BeoChip.C.

References writeByte().

#### **10.18.3.7 bool BeoChip::capturePulse (const int *whichone*, const bool *on*)**

Turn pulse captures on/off.

Definition at line 470 of file BeoChip.C.

References ASSERT, and writeByte().

**10.18.3.8 bool BeoChip::debounceKeyboard (const bool *on*)**

Turn keyboard debouncing on/off.

Definition at line 553 of file BeoChip.C.

References writeByte().

**10.18.3.9 bool BeoChip::debugMode (const bool *on*)**

Turn debug mode on/off.

When in debug mode, the BeoChip sends an EchoReply for each command it receives.

Definition at line 205 of file BeoChip.C.

References writeByte().

**10.18.3.10 bool BeoChip::echoRequest ()**

Send an echo request.

The BeoChip should reply with an EchoReply event if it is alive

Definition at line 198 of file BeoChip.C.

References writeByte().

**10.18.3.11 float BeoChip::getAnalog (const int *whichone*)**

Get current calibrated Analog value.

Definition at line 527 of file BeoChip.C.

References ASSERT, getAnalogRaw(), maxS, minS, rawToCalib(), and zeroS.

**10.18.3.12 byte BeoChip::getAnalogRaw (const int *whichone*)**

Get current raw Analog value.

Definition at line 536 of file BeoChip.C.

References adcval, and ASSERT.

Referenced by getAnalog().

**10.18.3.13 int BeoChip::getKeyboard ()**

Get current Keyboard value.

Definition at line 560 of file BeoChip.C.

References keyboard.

#### **10.18.3.14 float BeoChip::getPulse (const int *whichone*)**

Get current pulse value, between [-1.0..1.0].

Definition at line 489 of file BeoChip.C.

References ASSERT, getPulseRaw(), maxP, minP, rawToCalib(), and zeroP.

#### **10.18.3.15 short int BeoChip::getPulseRaw (const int *whichone*)**

Get current pulse value, raw uncalibrated.

Definition at line 498 of file BeoChip.C.

References ASSERT, and pulseval.

Referenced by getPulse().

#### **10.18.3.16 float BeoChip::getServo (const int *servo*) const**

Gets the current position of given servo.

Definition at line 437 of file BeoChip.C.

References ASSERT, getServoRaw(), maxS, minS, rawToCalib(), and zeroS.

#### **10.18.3.17 byte BeoChip::getServoRaw (const int *servo*) const**

Gets the current raw position of given servo.

Definition at line 463 of file BeoChip.C.

References ASSERT, and servopos.

Referenced by getServo().

#### **10.18.3.18 bool BeoChip::lcdClear ()**

Clear LCD screen.

Definition at line 283 of file BeoChip.C.

References lcdSendRaw().

**10.18.3.19 bool BeoChip::lcdCursorBlock ()**

Make cursor a blinking block.

Definition at line 325 of file BeoChip.C.

References lcdSendRaw().

**10.18.3.20 bool BeoChip::lcdCursorInvisible ()**

Make cursor invisible (not recommended).

Definition at line 337 of file BeoChip.C.

References lcdSendRaw().

**10.18.3.21 bool BeoChip::lcdCursorUnderline ()**

Make cursor a blinking underline.

Definition at line 331 of file BeoChip.C.

References lcdSendRaw().

**10.18.3.22 bool BeoChip::lcdGoto (const int i = 0)**

Go to given cursor position.

Definition at line 241 of file BeoChip.C.

References lcdSendRaw().

Referenced by lcdGotoXY().

**10.18.3.23 bool BeoChip::lcdGotoXY (const int x = 0, const int y = 0)**

Go to given cursor position.

Definition at line 219 of file BeoChip.C.

References ModelParam< int >::getVal(), itsLCDcols, itsLCDrows, and lcdGoto().

Referenced by lcdPrintf().

**10.18.3.24 bool BeoChip::lcdLoadFont (const [byte](#) data[64])**

Load a new font from an array of 64 bytes.

Definition at line 356 of file BeoChip.C.

References `lcdSendRaw()`.

#### **10.18.3.25 `bool BeoChip::lcdLoadFont (const int font)`**

Load one of the pre-programmed custom fonts.

Valid font values are [0..7]. Font 0 is an all-blank font.

Definition at line 343 of file `BeoChip.C`.

References `byte`, and `writeByte()`.

#### **10.18.3.26 `bool BeoChip::lcdMoveCursorLeft (const int i = 1)`**

Move cursor left by *i* positions.

Definition at line 307 of file `BeoChip.C`.

References `lcdSendRaw()`.

#### **10.18.3.27 `bool BeoChip::lcdMoveCursorRight (const int i = 1)`**

Move cursor right by *i* positions.

Definition at line 316 of file `BeoChip.C`.

References `lcdSendRaw()`.

#### **10.18.3.28 `bool BeoChip::lcdPrintf (const int x, const int y, const char * fmt, ...)`**

Print some text starting at given cursor position.

syntax is the same as `printf()`, plus the *x*, *y* coordinates. Returns true on success.

Definition at line 268 of file `BeoChip.C`.

References `ModelParam< int >::getVal()`, `itsLCDcols`, `lcdGotoXY()`, and `lcdPrintf()`.

#### **10.18.3.29 `bool BeoChip::lcdPrintf (const char * fmt, ...)`**

Print some text.

syntax is the same as `printf()`. Returns true on success.

Definition at line 251 of file `BeoChip.C`.

References `ModelParam< int >::getVal()`, `itsLCDcols`, and `lcdSendRaw()`.

Referenced by `lcdPrintf()`.



**10.18.3.30 bool BeoChip::lcdScrollLeft (const int *i* = 1)**

Scroll display left by *i* positions.

Definition at line 289 of file BeoChip.C.

References lcdSendRaw().

**10.18.3.31 bool BeoChip::lcdScrollRight (const int *i* = 1)**

Scroll display right by *i* positions.

Definition at line 298 of file BeoChip.C.

References lcdSendRaw().

**10.18.3.32 bool BeoChip::lcdSendRaw (const [byte](#) *val*, const bool *RS*)**

Send a raw byte to the LCD.

Definition at line 381 of file BeoChip.C.

References [byte](#), and writeByte().

Referenced by lcdClear(), lcdCursorBlock(), lcdCursorInvisible(), lcdCursorUnderline(), lcdGoto(), lcdLoadFont(), lcdMoveCursorLeft(), lcdMoveCursorRight(), lcdPrintf(), lcdScrollLeft(), and lcdScrollRight().

**10.18.3.33 bool BeoChip::lcdSetAnimation (const int *anim* = 0)**

Select an LCD animation and start it.

Valid anim values are [0..7]. Anim 0 is no animation.

Definition at line 368 of file BeoChip.C.

References [byte](#), and writeByte().

**10.18.3.34 float BeoChip::rawToCalib (const int *raw*, const int *zero*, const int *mini*, const int *maxi*) const [protected]**

Convert from raw (int) to calibrated (-1.0..1.0) position.

Definition at line 702 of file BeoChip.C.

Referenced by getAnalog(), getPulse(), getServo(), and run().

**10.18.3.35 bool BeoChip::resetChip ()**

Reset the BeoChip.

Definition at line 212 of file BeoChip.C.

References writeByte().

**10.18.3.36 void BeoChip::run ()**

This is our main running thread - don't call directly.

Should have been protected, but is not because of pthread hack.

Definition at line 592 of file BeoChip.C.

References adcval, ModelManager::debugMode(), SharedPtr< BeoChipListener >::get(), keyboard, maxA, maxP, minA, minP, pulseval, rawToCalib(), zeroA, and zeroP.

**10.18.3.37 bool BeoChip::setDigitalOut (const int *outnum*, const bool *on*)**

Turn a given digital output on/off.

Valid range for outnum is [0..3]

Definition at line 569 of file BeoChip.C.

References ASSERT, and writeByte().

**10.18.3.38 void BeoChip::setListener (SharedPtr< BeoChipListener > &  
*listener*)**

Install a callback (listener) for BeoChip events.

This callback will be called with the corresponding BeoChipEventType each time an event is received from the BeoChip.

Definition at line 194 of file BeoChip.C.

**10.18.3.39 bool BeoChip::setServo (const int *servo*, const float *position*)**

Moves servo # to given position in [-1.0 .. 1.0].

Returns true on success, false if some serial error occurred.

Definition at line 428 of file BeoChip.C.

References ASSERT, calibToRaw(), maxS, minS, setServoRaw(), and zeroS.

**10.18.3.40 bool BeoChip::setServoRaw (const int *servo*, const [byte](#) *val*)**

Sets servo number *servo* to value *val*.

Definition at line 447 of file BeoChip.C.

References ASSERT, servopos, and writeByte().

Referenced by setServo().

**10.18.3.41 bool BeoChip::shimServos (const [byte](#) *shim*)**

Shim the BeoChip's servo pulses.

Valid shim values are [0..7]. The default at bootup of the BeoChip is 4. Higher values will increase pulse length and lower values will decrease them. This may allow you to shim the pulses so as to maximally exploit the range of your servos.

Definition at line 405 of file BeoChip.C.

References writeByte().

**10.18.3.42 void BeoChip::start1 () [protected, virtual]**

open the port and get started

Reimplemented from [ModelComponent](#).

Definition at line 125 of file BeoChip.C.

References itsDevName.

**10.18.3.43 void BeoChip::stop2 () [protected, virtual]**

close the port and get stopped

Reimplemented from [ModelComponent](#).

Definition at line 161 of file BeoChip.C.

**10.18.3.44 bool BeoChip::writeByte (const [byte](#) *val*)**

send a byte to the BeoChip

You should never have to use this, use the other functions instead. This function is made public just for testing the chip's robustness against a flood of random junk hitting it... See test-BeoChip.C

Definition at line 577 of file BeoChip.C.

References `ModelManager::debugMode()`.

Referenced by `captureAnalog()`, `captureKeyboard()`, `capturePulse()`, `debounceKeyboard()`, `debugMode()`, `echoRequest()`, `lcdLoadFont()`, `lcdSendRaw()`, `lcdSetAnimation()`, `resetChip()`, `setDigitalOut()`, `setServoRaw()`, and `shimServos()`.

## 10.18.4 Member Data Documentation

### 10.18.4.1 `byte* BeoChip::adcval` [protected]

raw adc values [0..255]

Definition at line 326 of file `BeoChip.H`.

Referenced by `BeoChip()`, `getAnalogRaw()`, `run()`, and `~BeoChip()`.

### 10.18.4.2 `ModelParam<std::string> BeoChip::itsDevName` [protected]

name of our serial device

Definition at line 309 of file `BeoChip.H`.

Referenced by `start1()`.

### 10.18.4.3 `ModelParam<int> BeoChip::itsLCDcols` [protected]

number of columns of LCD screen

Definition at line 311 of file `BeoChip.H`.

Referenced by `lcdGotoXY()`, and `lcdPrintf()`.

### 10.18.4.4 `ModelParam<int> BeoChip::itsLCDrows` [protected]

number of rows of LCD screen (lines of text)

Definition at line 310 of file `BeoChip.H`.

Referenced by `lcdGotoXY()`.

### 10.18.4.5 `byte BeoChip::keyboard` [protected]

our current keyboard state

Definition at line 328 of file `BeoChip.H`.

Referenced by `BeoChip()`, `getKeyboard()`, and `run()`.

**10.18.4.6** `ModelParam<int>*` `BeoChip::maxA` [protected]

maximum raw value, for 1.0F (servo)

Definition at line 325 of file BeoChip.H.

Referenced by `BeoChip()`, `calibrateAnalog()`, `run()`, and `~BeoChip()`.

**10.18.4.7** `ModelParam<int>*` `BeoChip::maxP` [protected]

maximum raw value, for 1.0F (pulse)

Definition at line 320 of file BeoChip.H.

Referenced by `BeoChip()`, `calibratePulse()`, `getPulse()`, `run()`, and `~BeoChip()`.

**10.18.4.8** `ModelParam<int>*` `BeoChip::maxS` [protected]

maximum raw value, for 1.0F (servo)

Definition at line 315 of file BeoChip.H.

Referenced by `BeoChip()`, `calibrateServo()`, `getAnalog()`, `getServo()`, `setServo()`, and `~BeoChip()`.

**10.18.4.9** `ModelParam<int>*` `BeoChip::minA` [protected]

minimum raw value, for -1.0F (servo)

Definition at line 324 of file BeoChip.H.

Referenced by `BeoChip()`, `calibrateAnalog()`, `run()`, and `~BeoChip()`.

**10.18.4.10** `ModelParam<int>*` `BeoChip::minP` [protected]

minimum raw value, for -1.0F (pulse)

Definition at line 319 of file BeoChip.H.

Referenced by `BeoChip()`, `calibratePulse()`, `getPulse()`, `run()`, and `~BeoChip()`.

**10.18.4.11** `ModelParam<int>*` `BeoChip::minS` [protected]

minimum raw value, for -1.0F (servo)

Definition at line 314 of file BeoChip.H.

Referenced by `BeoChip()`, `calibrateServo()`, `getAnalog()`, `getServo()`, `setServo()`, and `~BeoChip()`.

#### 10.18.4.12 `short int* BeoChip::pulseval` [protected]

raw pulse values (11bit int)

Definition at line 321 of file `BeoChip.H`.

Referenced by `BeoChip()`, `getPulseRaw()`, `run()`, and `~BeoChip()`.

#### 10.18.4.13 `byte* BeoChip::servopos` [protected]

raw servo positions [0..255]

Definition at line 316 of file `BeoChip.H`.

Referenced by `BeoChip()`, `getServoRaw()`, `setServoRaw()`, and `~BeoChip()`.

#### 10.18.4.14 `ModelParam<int>* BeoChip::zeroA` [protected]

zero calibration value, for 0.0F (pulse)

Definition at line 323 of file `BeoChip.H`.

Referenced by `BeoChip()`, `calibrateAnalog()`, `run()`, and `~BeoChip()`.

#### 10.18.4.15 `ModelParam<int>* BeoChip::zeroP` [protected]

zero calibration value, for 0.0F (pulse)

Definition at line 318 of file `BeoChip.H`.

Referenced by `BeoChip()`, `calibratePulse()`, `getPulse()`, `run()`, and `~BeoChip()`.

#### 10.18.4.16 `ModelParam<int>* BeoChip::zeroS` [protected]

zero calibration value, for 0.0F (servo)

Definition at line 313 of file `BeoChip.H`.

Referenced by `BeoChip()`, `calibrateServo()`, `getAnalog()`, `getServo()`, `setServo()`, and `~BeoChip()`.

The documentation for this class was generated from the following files:

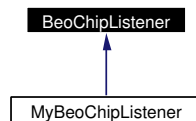
- [BeoChip.H](#)
- [BeoChip.C](#)

## 10.19 BeoChipListener Class Reference

BeoChip event listener.

```
#include <BeoChip.H>
```

Inheritance diagram for BeoChipListener:



### Public Member Functions

- virtual `~BeoChipListener()`  
*Destructor.*
- virtual void `event` (const `BeoChipEventType` t, const int valint, const float valfloat)=0  
*An event was received.*

### 10.19.1 Detailed Description

BeoChip event listener.

The `BeoChipListener` gets called each time an event is received from the `BeoChip`. The `BeoChipEventType` is passed to the listener, allowing the user to determine the type of event received and to then interpret the event data also passed to the listener. By default, there is no listener, and users can just asynchronously query the `BeoChip` for its current internal state. If those queries come in too slowly, however, they may miss some events (e.g., a briefly pressed key). So, production code should define a derived class for `BeoChipListener` and register it with the `BeoChip` object, so that it will take action as soon as an event is received.

Definition at line 63 of file `BeoChip.H`.

### 10.19.2 Constructor & Destructor Documentation

#### 10.19.2.1 `BeoChipListener::~BeoChipListener()` [virtual]

Destructor.

Definition at line 62 of file BeoChip.C.

### 10.19.3 Member Function Documentation

#### 10.19.3.1 virtual void BeoChipListener::event (const [BeoChipEventType](#) t, const int *valint*, const float *valfloat*) [pure virtual]

An event was received.

This function will be called by the [BeoChip](#) once you have registered your BeoChip-Listener derivative with the [BeoChip](#) using [BeoChip::setListener\(\)](#). Beware that the call to [event\(\)](#) will ve from a thread running in the [BeoChip](#) and thus will operate in parallel with your main thread and main loop. See test-BeoChip.C for an example of how to synchronize a main loop with these asynchronous event calls. The following data will be passed along with the event type:

| Type       | valint                     | valfloat                    |
|------------|----------------------------|-----------------------------|
| NONE       | 0                          | 0.0F                        |
| PWM0       | pulse 0 width, 1.6us units | calibrated width in [-1..1] |
| PWM1       | pulse 1 width, 1.6us units | calibrated width in [-1..1] |
| KBD        | 5 LSB show the 5 keys      | 0.0F                        |
| ADC0       | A/D 0 value                | calibrated value in [-1..1] |
| ADC1       | A/D 0 value                | calibrated value in [-1..1] |
| all others | 0                          | 0.0F                        |

Implemented in [MyBeoChipListener](#).

The documentation for this class was generated from the following files:

- [BeoChip.H](#)
- [BeoChip.C](#)

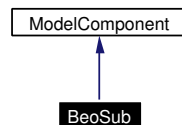


## 10.20 BeoSub Class Reference

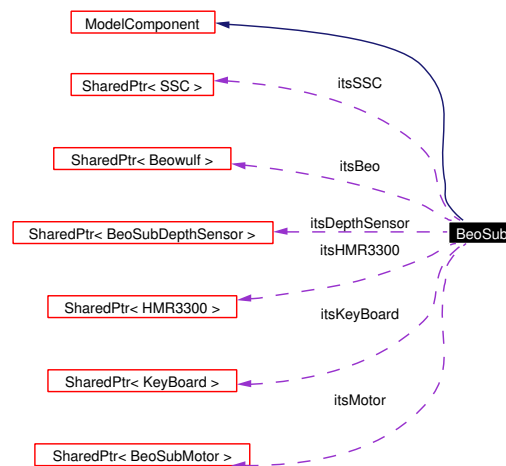
Definition and access functions for the BeoSub.

```
#include <BeoSub.H>
```

Inheritance diagram for BeoSub:



Collaboration diagram for BeoSub:



## Public Member Functions

### Constructors and Destructors

- [BeoSub](#) ([ModelManager](#) &mgr, const [std::string](#) &descrName="BeoSub", const [std::string](#) &tagName="BeoSub")

*Constructor.*

- [~BeoSub](#) ()

*Destructor.*

**Low-level access functions**

- float [getHeading](#) () const  
*Get current heading (yaw axis only).*
- float [getAveragedHeading](#) (const int n=20) const  
*Get current heading (yaw axis only), averaged over many measurements.*
- float [getDepth](#) () const  
*Get current depth.*
- float [getAveragedDepth](#) (const int n=20) const  
*Get current depth, averaged over many measurements.*
- int [getNbSlaves](#) () const  
*Get number of camera slaves.*
- bool [pulseMotor](#) (const int channel, const bool positive)  
*Send a pulse to a motor.*
- bool [turnOnMotor](#) (const int channel, const bool positive)  
*Turn on a motor at maximum speed.*
- bool [turnOffMotor](#) (const int channel)  
*Turn off a motor.*
- bool [moveMarkerGate](#) (const bool openclose)  
*Move marker gate.*
- [KeyBoardKey](#) [getKey](#) (const bool wait)  
*Get a key.*

**Low-level behaviors**

- bool [dive](#) (const float depth, const bool stabilize=true, const bool pulseWidthControl=false)  
*Dive to a specified depth.*
- bool [diveRel](#) (const float depth)  
*Relative Dive Hack.*
- bool [turn](#) (const float heading, const bool stabilize=true, const bool pulseWidthControl=false)  
*Turn to a specified heading.*
- bool [turnRel](#) (const float angle)

*Relative turn Hack.*

- bool [advance](#) (const int nsteps, const bool stabilize=true)  
*Advance for a specified number of length units.*
- bool [strafe](#) (const int nsteps, const float angle, const bool stabilize=true)  
*Strafe for a specified number of length units and direction.*
- [Image](#)< [PixRGB](#)< [byte](#) > > [grab](#) (const int node)  
*Grab a frame from slave grab-server process.*
- int [canny](#) (unsigned char \*[image](#), int rows, int cols, float sigma, float tlow, float thigh, unsigned char \*\*edge, char \*fname)
- void [gaussian\\_smooth](#) (unsigned char \*[image](#), int rows, int cols, float sigma, short int \*\*smoothedim)
- void [make\\_gaussian\\_kernel](#) (float sigma, float \*\*kernel, int \*windowsize)
- void [non\\_max\\_supp](#) (short \*mag, short \*gradx, short \*grady, int nrows, int ncols, unsigned char \*result)
- void [derrivative\\_x\\_y](#) (short int \*smoothedim, int rows, int cols, short int \*\*delta\_x, short int \*\*delta\_y)
- void [magnitude\\_x\\_y](#) (short int \*delta\_x, short int \*delta\_y, int rows, int cols, short int \*\*magnitude)
- int [follow\\_edges](#) (unsigned char \*edgemapptr, short \*edgemagptr, short lowval, int cols)
- int [apply\\_hysteresis](#) (short int \*mag, unsigned char \*nms, int rows, int cols, float tlow, float thigh, unsigned char \*edge)
- void [radian\\_direction](#) (short int \*delta\_x, short int \*delta\_y, int rows, int cols, float \*\*dir\_radians, int xdirtag, int ydirtag)
- double [angle\\_radians](#) (double x, double y)
- void [powell](#) (double p[ ], double \*\*xi, int n, double ftol, int \*iter, double \*fret, double(\*func)(double[ ]))
- void [checkpoint](#) (const char \*fmt,...)  
*Acquire and save a bunch of measurements.*

## Protected Attributes

- [SharedPtr](#)< [Beowulf](#) > [itsBeo](#)  
*Beowulf for communication with cameras.*
- [SharedPtr](#)< [HMR3300](#) > [itsHMR3300](#)  
*compass for heading*
- [SharedPtr](#)< [SSC](#) > [itsSSC](#)  
*ssc controls the marker drop gate*

- [SharedPtr](#)< [BeoSubMotor](#) > [itsMotor](#)  
*our motor plant*
- [SharedPtr](#)< [BeoSubDepthSensor](#) > [itsDepthSensor](#)  
*depth sensor*
- [SharedPtr](#)< [KeyBoard](#) > [itsKeyBoard](#)  
*for go signal*
- int [itsCkPt](#)  
*our checkpoint number*

### 10.20.1 Detailed Description

Definition and access functions for the BeoSub.

BeoSub is an autonomous visually-guided submarine. This class implements the high-level functionality, relying on low-level drivers to handle motor activation, compass reading, etc

Definition at line 67 of file BeoSub.H.

### 10.20.2 Constructor & Destructor Documentation

#### 10.20.2.1 BeoSub::BeoSub ([ModelManager](#) & *mgr*, const [std::string](#) & *descrName* = "BeoSub", const [std::string](#) & *tagName* = "BeoSub")

Constructor.

Definition at line 53 of file BeoSub.C.

References [ModelComponent::addSubComponent\(\)](#), [BEOSUB\\_PARPORT\\_ADDR](#), [BEOSUB\\_SERDEV\\_DEPTH](#), [BEOSUB\\_SERDEV\\_HMR3300](#), [BEOSUB\\_SERDEV\\_SSC](#), [itsBeo](#), [itsDepthSensor](#), [itsHMR3300](#), [itsMotor](#), and [itsSSC](#).

#### 10.20.2.2 BeoSub::~~BeoSub ()

Destructor.

Definition at line 75 of file BeoSub.C.

## 10.20.3 Member Function Documentation

### 10.20.3.1 **bool BeoSub::advance (const int *nsteps*, const bool *stabilize* = true)**

Advance for a specified number of length units.

**Parameters:**

*nsteps* number of steps, negative for reverse

*stabilize* if true, stabilize heading while diving

**Returns:**

true if successful, false if some problem occurred

Definition at line 223 of file BeoSub.C.

References ABS(), dive(), BeoSubAction::execute(), getDepth(), getHeading(), pulse-Motor(), and turn().

### 10.20.3.2 **void BeoSub::checkpoint (const char \* *fmt*, ...)**

Acquire and save a bunch of measurements.

Definition at line 288 of file BeoSub.C.

References getDepth(), getHeading(), getNbSlaves(), grab(), itsCkPt, and Raster::WriteRGB().

### 10.20.3.3 **bool BeoSub::dive (const float *depth*, const bool *stabilize* = true, const bool *pulseWidthControl* = false)**

Dive to a specified depth.

**Parameters:**

*depth* the desired depth

*stabilize* if true, stabilize heading while diving

**Returns:**

true if successful, false if some problem occurred

Definition at line 176 of file BeoSub.C.

References BeoSubAction::execute().

Referenced by advance().

**10.20.3.4 bool BeoSub::diveRel (const float *depth*)**

Relative Dive Hack.

Definition at line 207 of file BeoSub.C.

References pulseMotor().

**10.20.3.5 float BeoSub::getAveragedDepth (const int *n* = 20) const**

Get current depth, averaged over many measurements.

Definition at line 112 of file BeoSub.C.

References ASSERT, and getDepth().

**10.20.3.6 float BeoSub::getAveragedHeading (const int *n* = 20) const**

Get current heading (yaw axis only), averaged over many measurements.

Definition at line 95 of file BeoSub.C.

References ASSERT, and getHeading().

**10.20.3.7 float BeoSub::getDepth () const**

Get current depth.

Definition at line 106 of file BeoSub.C.

References SharedPtr< BeoSubDepthSensor >::get(), and itsDepthSensor.

Referenced by advance(), BeoSubActionTurn::BeoSubActionTurn(), checkpoint(), getAveragedDepth(), and BeoSubActionDive::getPosition().

**10.20.3.8 float BeoSub::getHeading () const**

Get current heading (yaw axis only).

Definition at line 79 of file BeoSub.C.

References SharedPtr< HMR3300 >::get(), Heading::getHead(), and itsHMR3300.

Referenced by advance(), BeoSubActionDive::BeoSubActionDive(), checkpoint(), getAveragedHeading(), and BeoSubActionTurn::getPosition().

**10.20.3.9    [KeyBoardKey](#) BeoSub::getKey (const bool *wait*)**

Get a key.

Definition at line 163 of file BeoSub.C.

References itsKeyBoard, and KeyBoardKey.

**10.20.3.10    int BeoSub::getNbSlaves () const**

Get number of camera slaves.

Definition at line 123 of file BeoSub.C.

References itsBeo.

Referenced by checkpoint(), and grab().

**10.20.3.11    [Image](#)< [PixRGB](#)< [byte](#) > > BeoSub::grab (const int *node*)**

Grab a frame from slave grab-server process.

**Parameters:**

*node* the [Beowulf](#) node number of the grab-server process

Definition at line 259 of file BeoSub.C.

References ASSERT, TCPmessage::getElement(), getNbSlaves(), and itsBeo.

Referenced by checkpoint().

**10.20.3.12    bool BeoSub::moveMarkerGate (const bool *openclose*)**

Move marker gate.

Definition at line 156 of file BeoSub.C.

References itsSSC.

**10.20.3.13    bool BeoSub::pulseMotor (const int *channel*, const bool *positive*)**

Send a pulse to a motor.

Definition at line 129 of file BeoSub.C.

References itsMotor.

Referenced by BeoSubActionTurn::activate(), BeoSubActionDive::activate(), advance(), diveRel(), and turnRel().

**10.20.3.14 bool BeoSub::strafe (const int *nsteps*, const float *angle*, const bool *stabilize* = true)**

Strafe for a specified number of length units and direction.

**Parameters:**

*nsteps* number of steps, should always be given positive

*angle* angluar deviation in degrees

*stabilize* if true, stabilize heading while diving

**Returns:**

true if successful, false if some problem occurred

Definition at line 248 of file BeoSub.C.

**10.20.3.15 bool BeoSub::turn (const float *heading*, const bool *stabilize* = true, const bool *pulseWidthControl* = false)**

Turn to a specified heading.

**Parameters:**

*heading* the desired heading (yaw axis only)

*stabilize* if true, stabilize heading while diving

**Returns:**

true if successful, false if some problem occurred

Definition at line 183 of file BeoSub.C.

References BeoSubAction::execute().

Referenced by advance().

**10.20.3.16 bool BeoSub::turnOffMotor (const int *channel*)**

Turn off a motor.

Definition at line 148 of file BeoSub.C.

References itsMotor.

**10.20.3.17 bool BeoSub::turnOnMotor (const int *channel*, const bool *positive*)**

Turn on a motor at maximum speed.



Definition at line 135 of file BeoSub.C.

References itsMotor.

#### 10.20.3.18 bool BeoSub::turnRel (const float *angle*)

Relative turn Hack.

Definition at line 191 of file BeoSub.C.

References pulseMotor().

### 10.20.4 Member Data Documentation

#### 10.20.4.1 [SharedPtr<Beowulf>](#) BeoSub::itsBeo [protected]

[Beowulf](#) for communication with cameras.

Definition at line 187 of file BeoSub.H.

Referenced by BeoSub(), getNbSlaves(), and grab().

#### 10.20.4.2 int BeoSub::itsCkPt [protected]

our checkpoint number

Definition at line 194 of file BeoSub.H.

Referenced by checkpoint().

#### 10.20.4.3 [SharedPtr<BeoSubDepthSensor>](#) BeoSub::itsDepthSensor [protected]

depth sensor

Definition at line 191 of file BeoSub.H.

Referenced by BeoSub(), and getDepth().

#### 10.20.4.4 [SharedPtr<HMR3300>](#) BeoSub::itsHMR3300 [protected]

compass for heading

Definition at line 188 of file BeoSub.H.

Referenced by BeoSub(), and getHeading().

**10.20.4.5**   [SharedPtr<Keyboard>](#) [BeoSub::itsKeyboard](#)   [protected]

for go signal

Definition at line 192 of file BeoSub.H.

Referenced by getKey().

**10.20.4.6**   [SharedPtr<BeoSubMotor>](#) [BeoSub::itsMotor](#)   [protected]

our motor plant

Definition at line 190 of file BeoSub.H.

Referenced by BeoSub(), pulseMotor(), turnOffMotor(), and turnOnMotor().

**10.20.4.7**   [SharedPtr<SSC>](#) [BeoSub::itsSSC](#)   [protected]

ssc controls the marker drop gate

Definition at line 189 of file BeoSub.H.

Referenced by BeoSub(), and moveMarkerGate().

The documentation for this class was generated from the following files:

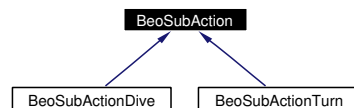
- [BeoSub.H](#)
- [BeoSub.C](#)

## 10.21 BeoSubAction Class Reference

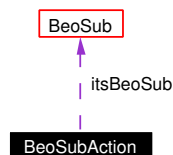
Helper base class for [BeoSub](#) motor actions.

```
#include <BeoSubAction.H>
```

Inheritance diagram for BeoSubAction:



Collaboration diagram for BeoSubAction:



### Public Member Functions

- `BeoSubAction (BeoSub *bs, const bool pulseWidthControl=false)`  
*Constructor.*
- `virtual ~BeoSubAction ()`  
*Destructor.*
- `virtual bool execute (const float target, const bool stabil, const int itsMaxIter=100)`  
*Execute the action.*
- `virtual bool pulseWidthExecute (const float target, const bool stabil, const int itsMaxIter=100)`

### Protected Member Functions

- `virtual bool activate (const bool incr)=0`  
*Activate the action for one step.*

- virtual bool **turnOnMotor** (const bool incr)=0
- virtual bool **turnOffMotor** (const bool incr)=0
- virtual float **getPosition** () const=0  
*Get a reading of the current position.*
- virtual bool **stabilize** (const bool **pulseWidthControl**=false)=0  
*Stabilize whatever is not being modified by the action.*
- virtual float **difference** (float pos, float target)=0  
*Calculate current and target value differences.*

## Protected Attributes

- **BeoSub** \* **itsBeoSub**
- float **itsDeadBand**  
*our **BeoSub**, which holds our actuators & instruments*
- bool **pulseWidthControl**  
*deadband*
- float **itsGainP**  
*true when using pulse width control, number of pulse control when false*
- float **itsGainD**  
*Gain for Propositional Control.*
- float **itsGainI**  
*Gain for Derivative Control.*
- int **turnOnTime**  
*Gain for Integral Control.*
- float **itsErrorHistory** [20]  
*Number of time steps (iterations) for the motor to turn on when going pulse interval control.*
- int **itsErrIndex**  
*storing pass 20 iterations value*
- float **u**  
*index top iterate the history*

### 10.21.1 Detailed Description

Helper base class for [BeoSub](#) motor actions.

The base class defines an interface that will then be used in various derived forms in order to perform a variety of actions

Definition at line 46 of file BeoSubAction.H.

### 10.21.2 Constructor & Destructor Documentation

#### 10.21.2.1 **BeoSubAction::BeoSubAction** ([BeoSub](#) \* *bs*, const bool *pulseWidthControl* = false)

Constructor.

Definition at line 44 of file BeoSubAction.C.

#### 10.21.2.2 **BeoSubAction::~BeoSubAction** () [virtual]

Destructor.

Definition at line 49 of file BeoSubAction.C.

### 10.21.3 Member Function Documentation

#### 10.21.3.1 **virtual bool BeoSubAction::activate** (const bool *incr*) [protected, pure virtual]

Activate the action for one step.

**Parameters:**

*incr* if true, the one step of activation should tend to increase the value returned by [getPosition\(\)](#), otherwise it should tend to decrease it

**Returns:**

true on success

Implemented in [BeoSubActionDive](#), and [BeoSubActionTurn](#).

Referenced by [execute\(\)](#).

#### 10.21.3.2 **virtual float BeoSubAction::difference** (float *pos*, float *target*) [protected, pure virtual]

Calculate current and target value differences.

Implemented in [BeoSubActionDive](#), and [BeoSubActionTurn](#).

Referenced by `execute()`.

#### 10.21.3.3 **bool BeoSubAction::execute (const float *target*, const bool *stabil*, const int *itsMaxIter* = 100) [virtual]**

Execute the action.

This will attempt to reach the target reading by iteratively outputting motor commands and monitoring the instruments. If *stabil* is true, we will attempt to maintain the sub stable while executing the action. Returns true on success.

Definition at line 53 of file `BeoSubAction.C`.

References `activate()`, `difference()`, `getPosition()`, `itsDeadBand`, `itsErrIndex`, `itsErrorHistory`, `itsGainD`, `itsGainI`, `itsGainP`, `pulseWidthControl`, `stabilize()`, and `u`.

Referenced by `BeoSub::advance()`, `BeoSub::dive()`, `BeoSubActionTurn::stabilize()`, `BeoSubActionDive::stabilize()`, and `BeoSub::turn()`.

#### 10.21.3.4 **virtual float BeoSubAction::getPosition () const [protected, pure virtual]**

Get a reading of the current position.

Implemented in [BeoSubActionDive](#), and [BeoSubActionTurn](#).

Referenced by `execute()`.

#### 10.21.3.5 **virtual bool BeoSubAction::stabilize (const bool *pulseWidthControl* = false) [protected, pure virtual]**

Stabilize whatever is not being modified by the action.

Implemented in [BeoSubActionDive](#), and [BeoSubActionTurn](#).

Referenced by `execute()`.

### 10.21.4 Member Data Documentation

#### 10.21.4.1 **float [BeoSubAction::itsDeadBand](#) [protected]**

our [BeoSub](#), which holds our actuators & instruments

Definition at line 89 of file `BeoSubAction.H`.

Referenced by `execute()`.

**10.21.4.2 int [BeoSubAction::itsErrIndex](#) [protected]**

storing pass 20 iterations value

Definition at line 99 of file BeoSubAction.H.

Referenced by execute().

**10.21.4.3 float [BeoSubAction::itsErrorHistory](#)[20] [protected]**

Number of time steps (iterations) for the motor to turn on when going pulse interval control.

Definition at line 98 of file BeoSubAction.H.

Referenced by execute().

**10.21.4.4 float [BeoSubAction::itsGainD](#) [protected]**

Gain for Propositional Control.

Definition at line 93 of file BeoSubAction.H.

Referenced by execute().

**10.21.4.5 float [BeoSubAction::itsGainI](#) [protected]**

Gain for Derivative Control.

Definition at line 94 of file BeoSubAction.H.

Referenced by execute().

**10.21.4.6 float [BeoSubAction::itsGainP](#) [protected]**

true when using pulse width control, number of pulse control when false

Definition at line 92 of file BeoSubAction.H.

Referenced by execute().

**10.21.4.7 bool [BeoSubAction::pulseWidthControl](#) [protected]**

deadband

Definition at line 90 of file BeoSubAction.H.

Referenced by execute().

**10.21.4.8**   **int** [BeoSubAction::turnOnTime](#)   [protected]

Gain for Integral Control.

Definition at line 96 of file BeoSubAction.H.

**10.21.4.9**   **float** [BeoSubAction::u](#)   [protected]

index top iterate the history

Definition at line 100 of file BeoSubAction.H.

Referenced by execute().

The documentation for this class was generated from the following files:

- [BeoSubAction.H](#)
- [BeoSubAction.C](#)

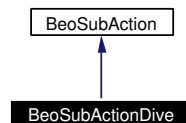


## 10.22 BeoSubActionDive Class Reference

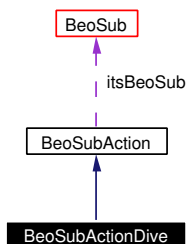
Helper base class for [BeoSub](#) dive actions.

```
#include <BeoSubAction.H>
```

Inheritance diagram for BeoSubActionDive:



Collaboration diagram for BeoSubActionDive:



### Public Member Functions

- [BeoSubActionDive](#) ([BeoSub](#) \*bs, bool [pulseWidthControl](#)=false)  
*Constructor.*
- virtual [~BeoSubActionDive](#) ()  
*Destructor.*

### Protected Member Functions

- virtual bool [activate](#) (const bool incr)  
*Dive for one step (go deeper if incr=true).*
- virtual bool **turnOnMotor** (const bool incr)
- virtual bool **turnOffMotor** (const bool incr)

- virtual float [getPosition](#) () const  
*Get a reading of the current depth.*
- virtual bool [stabilize](#) (const bool [pulseWidthControl](#)=false)  
*Stabilize the heading.*
- virtual float [difference](#) (float pos, float target)  
*Calculate current and target value differences.*

### 10.22.1 Detailed Description

Helper base class for [BeoSub](#) dive actions.

Definition at line 105 of file BeoSubAction.H.

### 10.22.2 Constructor & Destructor Documentation

#### 10.22.2.1 [BeoSubActionDive::BeoSubActionDive](#) ([BeoSub](#) \* *bs*, bool *pulseWidthControl* = false)

Constructor.

Definition at line 176 of file BeoSubAction.C.

References [BeoSub::getHeading\(\)](#).

#### 10.22.2.2 [BeoSubActionDive::~~BeoSubActionDive](#) () [virtual]

Destructor.

Definition at line 195 of file BeoSubAction.C.

### 10.22.3 Member Function Documentation

#### 10.22.3.1 [bool BeoSubActionDive::activate](#) (const bool *incr*) [protected, virtual]

Dive for one step (go deeper if incr=true).

Implements [BeoSubAction](#).

Definition at line 199 of file BeoSubAction.C.

References [BeoSub::pulseMotor\(\)](#).

**10.22.3.2 float BeoSubActionDive::difference (float *pos*, float *target*)**  
[protected, virtual]

Calculate current and target value differences.

Implements [BeoSubAction](#).

Definition at line 231 of file BeoSubAction.C.

**10.22.3.3 float BeoSubActionDive::getPosition () const** [protected, virtual]

Get a reading of the current depth.

Implements [BeoSubAction](#).

Definition at line 219 of file BeoSubAction.C.

References [BeoSub::getDepth\(\)](#).

**10.22.3.4 bool BeoSubActionDive::stabilize (const bool *pulseWidthControl* = false)** [protected, virtual]

Stabilize the heading.

Implements [BeoSubAction](#).

Definition at line 223 of file BeoSubAction.C.

References [BeoSubAction::execute\(\)](#).

The documentation for this class was generated from the following files:

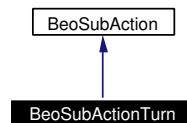
- [BeoSubAction.H](#)
- [BeoSubAction.C](#)

## 10.23 BeoSubActionTurn Class Reference

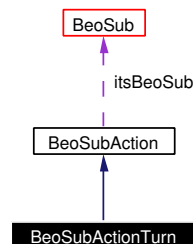
Helper base class for [BeoSub](#) turn actions.

```
#include <BeoSubAction.H>
```

Inheritance diagram for BeoSubActionTurn:



Collaboration diagram for BeoSubActionTurn:



### Public Member Functions

- [BeoSubActionTurn](#) ([BeoSub](#) \*bs, bool [pulseWidthControl](#)=false)  
*Constructor.*
- virtual [~BeoSubActionTurn](#) ()  
*Destructor.*

### Protected Member Functions

- virtual bool [activate](#) (const bool incr)  
*Turn for one step (turn right if incr=true).*
- virtual bool **turnOnMotor** (const bool incr)
- virtual bool **turnOffMotor** (const bool incr)

- virtual float [getPosition](#) () const  
*Get a reading of the current heading.*
- virtual bool [stabilize](#) (const bool [pulseWidthControl](#)=false)  
*Stabilize the depth.*
- virtual float [difference](#) (float pos, float target)

### 10.23.1 Detailed Description

Helper base class for [BeoSub](#) turn actions.

Definition at line 141 of file BeoSubAction.H.

### 10.23.2 Constructor & Destructor Documentation

#### 10.23.2.1 BeoSubActionTurn::BeoSubActionTurn ([BeoSub](#) \* *bs*, bool *pulseWidthControl* = false)

Constructor.

Definition at line 240 of file BeoSubAction.C.

References [BeoSub::getDepth\(\)](#).

#### 10.23.2.2 BeoSubActionTurn::~~BeoSubActionTurn () [virtual]

Destructor.

Definition at line 260 of file BeoSubAction.C.

### 10.23.3 Member Function Documentation

#### 10.23.3.1 bool BeoSubActionTurn::activate (const bool *incr*) [protected, virtual]

Turn for one step (turn right if *incr*=true).

Implements [BeoSubAction](#).

Definition at line 264 of file BeoSubAction.C.

References [BeoSub::pulseMotor\(\)](#).

#### 10.23.3.2 **float BeoSubActionTurn::difference (float *pos*, float *target*)** [protected, virtual]

Calculate current and target value differences Returning shortest turning angle with clockwise direction (positive)

Implements [BeoSubAction](#).

Definition at line 294 of file BeoSubAction.C.

#### 10.23.3.3 **float BeoSubActionTurn::getPosition () const** [protected, virtual]

Get a reading of the current heading.

Implements [BeoSubAction](#).

Definition at line 283 of file BeoSubAction.C.

References [BeoSub::getHeading\(\)](#).

#### 10.23.3.4 **bool BeoSubActionTurn::stabilize (const bool *pulseWidthControl* = false)** [protected, virtual]

Stabilize the depth.

Implements [BeoSubAction](#).

Definition at line 287 of file BeoSubAction.C.

References [BeoSubAction::execute\(\)](#).

The documentation for this class was generated from the following files:

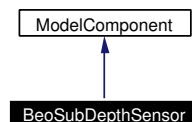
- [BeoSubAction.H](#)
- [BeoSubAction.C](#)

## 10.24 BeoSubDepthSensor Class Reference

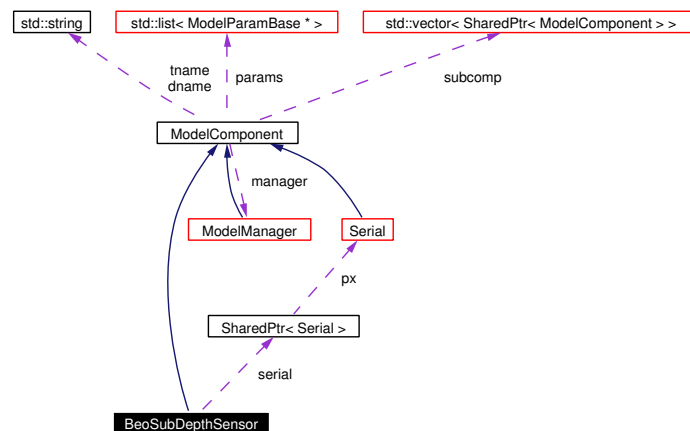
Class for interfacing with a depth sensor.

```
#include <BeoSubDepthSensor.H>
```

Inheritance diagram for BeoSubDepthSensor:



Collaboration diagram for BeoSubDepthSensor:



### Public Member Functions

- [BeoSubDepthSensor](#) ([ModelManager](#) &mgr, const [std::string](#) &descr-Name="Depth Sensor", const [std::string](#) &tagName="DepthSensor", const char \*dev="/dev/ttyS0")  
*Initialize.*
- [~BeoSubDepthSensor](#) ()  
*Destructor.*
- float [get](#) ()  
*read the depth*

- void [run](#) ()

*The user need not concern himself with this method.*

## Protected Member Functions

- void [start2](#) ()

*get started*

- void [stop1](#) ()

*get stopped*

### 10.24.1 Detailed Description

Class for interfacing with a depth sensor.

Definition at line 46 of file BeoSubDepthSensor.H.

### 10.24.2 Constructor & Destructor Documentation

#### 10.24.2.1 BeoSubDepthSensor::BeoSubDepthSensor ([ModelManager](#) & mgr, const [std::string](#) & descrName = "Depth Sensor", const [std::string](#) & tagName = "DepthSensor", const char \* dev = "/dev/ttyS0")

Initialize.

Definition at line 52 of file BeoSubDepthSensor.C.

References [ModelComponent::addSubComponent\(\)](#).

#### 10.24.2.2 BeoSubDepthSensor::~~BeoSubDepthSensor ()

Destructor.

Definition at line 77 of file BeoSubDepthSensor.C.

### 10.24.3 Member Function Documentation

#### 10.24.3.1 float BeoSubDepthSensor::get ()

read the depth



Definition at line 109 of file BeoSubDepthSensor.C.

#### 10.24.3.2 void BeoSubDepthSensor::run ()

The user need not concern himself with this method.

Should have been protected, but is not because of the pthread hack.

Definition at line 81 of file BeoSubDepthSensor.C.

#### 10.24.3.3 void BeoSubDepthSensor::start2 () [protected, virtual]

get started

Reimplemented from [ModelComponent](#).

Definition at line 66 of file BeoSubDepthSensor.C.

#### 10.24.3.4 void BeoSubDepthSensor::stop1 () [protected, virtual]

get stopped

Reimplemented from [ModelComponent](#).

Definition at line 70 of file BeoSubDepthSensor.C.

The documentation for this class was generated from the following files:

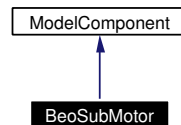
- [BeoSubDepthSensor.H](#)
- [BeoSubDepthSensor.C](#)

## 10.25 BeoSubMotor Class Reference

Low-level driver for the [BeoSub](#) motors.

```
#include <BeoSubMotor.H>
```

Inheritance diagram for BeoSubMotor:



Collaboration diagram for BeoSubMotor:

### Public Member Functions

- **BeoSubMotor** ([ModelManager](#) &mgr, const [std::string](#) &descrName="BeoSub Motor", const [std::string](#) &tagName="BeoSub Motor", const int def\_parallel\_port\_addr=0x378)
- [~BeoSubMotor](#) ()  
*Destructor.*
- bool [setValue](#) (const [byte](#) value, const int channel, const bool immed=true)  
*Set a value.*
- [byte](#) [getValue](#) (const int channel) const  
*Get the last set value on a given channel.*
- bool [pulseValue](#) (const int channel, const bool positive)  
*Pulse a channel on for a bit.*
- bool [WritePort](#) ()  
*Write out our internal data to the parallel port.*

### Public Attributes

- unsigned char **chMin** [9]
- unsigned char **chMax** [9]
- unsigned char **chDefault** [9]

## Protected Member Functions

- void **reset** ()
- void **SendCh** (const int value)

## Protected Attributes

- [ModelParam](#)< int > [itsPort](#)  
*base address of parallel port*
- unsigned char **strobelo**
- unsigned char **strobeh**
- unsigned char [ch](#) [9]  
*channel values*

### 10.25.1 Detailed Description

Low-level driver for the [BeoSub](#) motors.

Definition at line 63 of file BeoSubMotor.H.

### 10.25.2 Constructor & Destructor Documentation

#### 10.25.2.1 [BeoSubMotor::~BeoSubMotor](#) ()

Destructor.

Definition at line 81 of file BeoSubMotor.C.

References [ModelComponent::reset\(\)](#).

### 10.25.3 Member Function Documentation

#### 10.25.3.1 [byte](#) [BeoSubMotor::getValue](#) (const int *channel*) const

Get the last set value on a given channel.

Definition at line 169 of file BeoSubMotor.C.

References [ASSERT](#), and [ch](#).

**10.25.3.2 bool BeoSubMotor::pulseValue (const int *channel*, const bool *positive*)**

Pulse a channel on for a bit.

**Parameters:**

***positive*** use a pulse value higher than rest if true, lower then rest otherwise

Definition at line 138 of file BeoSubMotor.C.

References ASSERT, and setValue().

**10.25.3.3 bool BeoSubMotor::setValue (const [byte](#) *value*, const int *channel*, const bool *immed* = true)**

Set a value.

Set a value not less than a minimum and not larger than maximum to protect controller box from damage write the value to the parallel port

**Parameters:**

***value*** the value to set

***channel*** the channel to set the value to

***immed*** if true, immediately send all values to the interface, otherwise just store the value internally and it will be sent at the next writePort() or [setValue\(\)](#) with immed set to true.

**Returns:**

true on success

Definition at line 124 of file BeoSubMotor.C.

References ASSERT, ch, and WritePort().

Referenced by pulseValue().

**10.25.3.4 bool BeoSubMotor::WritePort ()**

Write out our internal data to the parallel port.

**Returns:**

true on success

Definition at line 87 of file BeoSubMotor.C.

References ch, ModelParam< int >::getVal(), and itsPort.

Referenced by setValue().

## 10.25.4 Member Data Documentation

### 10.25.4.1 unsigned char [BeoSubMotor::ch](#)[9] [protected]

channel values

Definition at line 107 of file BeoSubMotor.H.

Referenced by [getValue\(\)](#), [setValue\(\)](#), and [WritePort\(\)](#).

### 10.25.4.2 [ModelParam](#)<int> [BeoSubMotor::itsPort](#) [protected]

base address of parallel port

Definition at line 104 of file BeoSubMotor.H.

Referenced by [WritePort\(\)](#).

The documentation for this class was generated from the following files:

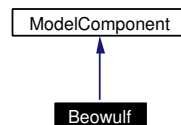
- [BeoSubMotor.H](#)
- [BeoSubMotor.C](#)

## 10.26 Beowulf Class Reference

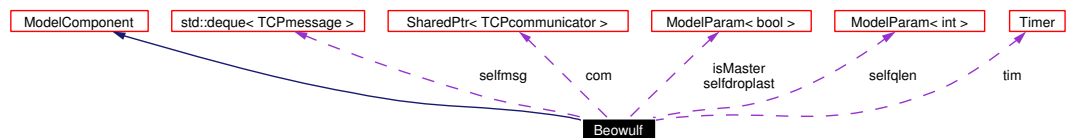
Simple interfacing to a Beowulf cluster.

```
#include <Beowulf.H>
```

Inheritance diagram for Beowulf:



Collaboration diagram for Beowulf:



### Public Member Functions

#### Constructors and destructors

- **Beowulf** (**ModelManager** &mgr, const **std::string** &descrName="Beowulf", const **std::string** &tagName="Beowulf", const bool ismaster=false)

*Constructor.*

- virtual void **exportOptions** (const int whatToExport, const bool recurse=true)

*Export some of our **ModelParam** parameters as command-line-options.*

- void **resetConnections** (const int keepfd=-1)

*Reset and kill all connections except possibly one (keepfd).*

- virtual **~Beowulf** ()

*Destructor.*

#### Access functions

- int **getNbSlaves** () const

*get number of slave nodes*

- int [getNodeNumber](#) () const  
*get our node number (-1 is the master)*
- const char \* [nodeName](#) (const int nb) const  
*Get hostname:port of node with given node number.*
- int [requestNode](#) ()  
*Request a so-far unallocated node.*

### Message passing functions

- void [send](#) (const int node\_nb, [TCPmessage](#) &msg)  
*Send message to another node.*
- void [send](#) ([TCPmessage](#) &msg)  
*Send message to the least-loaded of our slave nodes.*
- bool [receive](#) (int &node\_nb, [TCPmessage](#) &msg, [int32](#) &frame, [int32](#) &action, const int timeout=0)  
*Receive message from a given node (or from any node).*
- int [nbReceived](#) (const int node\_nb=-2)  
*Do we have any received messages?*

### Protected Member Functions

- virtual bool [setModelParamString](#) (const [std::string](#) &name, const [std::string](#) &value, const bool recurse=false, const bool warn=true)  
*Intercept people changing our [ModelParam](#).*

### Protected Attributes

- [ModelParam](#)< [std::string](#) > [itsSlaveNames](#)  
*names of our slaves as a space-separated list of hostname:port*
- [ModelParam](#)< bool > [isMaster](#)  
*true if we are the master*
- [ModelParam](#)< int > [selfqlen](#)

*self-message queue length*

- `ModelParam` < bool > `selfdroplast`

*self-message queue drop policy*

### 10.26.1 Detailed Description

Simple interfacing to a Beowulf cluster.

The idea of this class is to hide all of the low-level communication setup and transfer details from the user, and to provide a simple interface for passing messages between nodes on a Beowulf cluster. Each slave node should instantiate a Beowulf object and initialize it with `slaveInit()`. This will block the slave until it is contacted by the Beowulf master node. The master node instantiates a Beowulf object, and initializes it using `masterInit()`, passing along the hostnames of the slave nodes. During initialization, the Beowulf master contacts all the slaves and instructs them to fully interconnect with each other. Once initialization is complete, any node can `send()` and `receive()` TCPmessages to and from any other node. Both `send()` and `receive()` are non-blocking methods. Actual queueing and transfer of messages is done in a thread that runs in parallel with the main program thread.

Definition at line 72 of file Beowulf.H.

### 10.26.2 Constructor & Destructor Documentation

**10.26.2.1** `Beowulf::Beowulf (ModelManager & mgr, const std::string & descrName = "Beowulf", const std::string & tagName = "Beowulf", const bool ismaster = false)`

Constructor.

**Parameters:**

***isMaster*** true if we are the master of the Beowulf. The master is the one that gets the list of slaves and then initializes all the slaves at `start()` time.

Definition at line 44 of file Beowulf.C.

References `ModelComponent::addSubComponent()`.

**10.26.2.2** `Beowulf::~Beowulf ()` [virtual]

Destructor.

Will properly terminate all connections.



Definition at line 65 of file Beowulf.C.

### 10.26.3 Member Function Documentation

#### 10.26.3.1 void Beowulf::exportOptions (const int *whatToExport*, const bool *recurse* = true) [virtual]

Export some of our [ModelParam](#) parameters as command-line-options.

See the base function in [ModelComponent.H](#) for info.

Reimplemented from [ModelComponent](#).

Definition at line 69 of file Beowulf.C.

References [ModelComponent::exportOptions\(\)](#), [ModelParam< bool >::getVal\(\)](#), [is-Master](#), [itsSlaveNames](#), [OPTEXP\\_CORE](#), [ModelManager::requestOption\(\)](#), [self-droplast](#), and [selfqlen](#).

#### 10.26.3.2 int Beowulf::getNbSlaves () const

get number of slave nodes

Definition at line 117 of file Beowulf.C.

#### 10.26.3.3 int Beowulf::getNodeNumber () const

get our node number (-1 is the master)

Definition at line 121 of file Beowulf.C.

#### 10.26.3.4 int Beowulf::nbReceived (const int *node\_nb* = -2)

Do we have any received messages?

Returns the total number of messages in the incoming queues of our various connected nodes. If *node\_nb* == -1, only consider messages from the Beowulf master. If *node\_nb* == -2, consider any node, otherwise only consider the specified node.

Definition at line 577 of file Beowulf.C.

#### 10.26.3.5 const char \* Beowulf::nodeName (const int *nb*) const

Get hostname:port of node with given node number.

This is whatever the user gave at configuration, so it could be just a short hostname, a fully-qualified hostname, or a hostname:port. If *nb* is -1, we return "BeoMaster"

Definition at line 125 of file Beowulf.C.

#### 10.26.3.6 **bool Beowulf::receive (int & node\_nb, TCPmessage & msg, int32 & frame, int32 & action, const int timeout = 0)**

Receive message from a given node (or from any node).

Check whether a message has been received; returns false otherwise. If a message was received, the node it came from will be in node\_nb, and its frame and action fields will be pre-decoded for convenience (they still are in the message itself too). This method is always non-blocking, i.e., it returns immediately and does not wait for messages to come in.

##### Parameters:

**node\_nb** node number to receive from, or -1 to receive from any node. If a message is received, node\_nb will be updated to the node number from which the message was received, or -1 if it was received from the master.

**msg** received message

**frame** frame number from received message

**action** action field from received message

**timeout** if non-zero, max time (in ms) this call may block

Definition at line 511 of file Beowulf.C.

References ASSERT, TCPmessage::getAction(), TCPmessage::getETI(), TCPmessage::getID(), and Timer::getSecs().

#### 10.26.3.7 **int Beowulf::requestNode ()**

Request a so-far unallocated node.

This will return the next node number that has not yet been requested. This only works if we are Beowulf master. It will generate an error message and return -2 if we have no more unallocated nodes. We need to be [start\(\)](#)'ed for this to work.

Definition at line 135 of file Beowulf.C.

References ModelComponent::started().

#### 10.26.3.8 **void Beowulf::resetConnections (const int keepfd = -1)**

Reset and kill all connections except possibly one (keepfd).

Resets the Beowulf to uninitialized state. Kills all connections, except possibly one (typically towards the master) that may be specified as argument.

**Parameters:**

*keepfd* the fd to keep (or -1 to keep no fd)

Definition at line 145 of file Beowulf.C.

References `Timer::reset()`.

**10.26.3.9 void Beowulf::send (TCPmessage & msg)**

Send message to the least-loaded of our slave nodes.

This method is non-blocking (returns immediately). A copy of msg is taken, so you can destroy it immediately after send. Only works if we are the Beowulf master, fatal error otherwise. This implements load balancing. The ETI (estimated time to idle) fields in `TCPmessage` are used to determine which of our slave nodes has the shortest pending work queue (i.e., shortest ETI) and the message will be sent to that node. Thus, this functionality assumes that every slave node can process every message that you might send them (as opposed to more constrained architectures where a given node is only capable of doing a given type of processing corresponding to a given type of received message). For this load balancing to work, the slaves should try to put good-faith estimates of their time to idle (in seconds) each time they send us (the master) a message back. The master relies on those good-faith estimates to decide which node is the least loaded. This approach has severe limitations if your overall message traffic is low, as your ETI estimates at the master will not be refreshed regularly and may become grossly inaccurate. Thus, this approach is mostly intended for streaming applications, where every node will usually send several messages back to the master every 30ms or so, so that the ETI estimates collected at the master will be reasonably fresh and accurate. If several slave nodes have the lowest ETI, one will be picked at random.

Definition at line 475 of file Beowulf.C.

References `ASSERT`, `TCPmessage::getAction()`, and `TCPmessage::getID()`.

**10.26.3.10 void Beowulf::send (const int node\_nb, TCPmessage & msg)**

Send message to another node.

This method is non-blocking (returns immediately). A copy of msg is taken, so you can destroy it immediately after send.

**Parameters:**

*node\_nb* is the destination node number. A value of -1 on a slave Beowulf means that msg should be sent to the Beowulf master.

Definition at line 434 of file Beowulf.C.

References `ASSERT`, `TCPmessage::getAction()`, `TCPmessage::getID()`, `ModelParam< bool >::getVal()`, `ModelParam< int >::getVal()`, `selfdroplast`, and `selfqlen`.

**10.26.3.11** `bool Beowulf::setModelParamString (const std::string & name, const std::string & value, const bool recurse = false, const bool warn = true)` [protected, virtual]

Intercept people changing our [ModelParam](#).

See [ModelComponent.H](#); as parsing the command-line or reading a config file sets our name, we'll also here instantiate a controller of the proper type (and export its options)

Reimplemented from [ModelComponent](#).

Definition at line 84 of file Beowulf.C.

References [ModelParamBase::getName\(\)](#), [ModelParam< bool >::getVal\(\)](#), [isMaster](#), [itsSlaveNames](#), [ModelManager::requestOption\(\)](#), and [ModelComponent::setModelParamString\(\)](#).

## 10.26.4 Member Data Documentation

**10.26.4.1** `ModelParam<bool> Beowulf::isMaster` [protected]

true if we are the master

Definition at line 202 of file Beowulf.H.

Referenced by [exportOptions\(\)](#), and [setModelParamString\(\)](#).

**10.26.4.2** `ModelParam<std::string> Beowulf::itsSlaveNames` [protected]

names of our slaves as a space-separated list of hostname:port

port is optional and we will use the default [SockServ](#) port if unspecified. This parameter is only used if we are Beowulf master (see constructor)

Definition at line 200 of file Beowulf.H.

Referenced by [exportOptions\(\)](#), and [setModelParamString\(\)](#).

**10.26.4.3** `ModelParam<bool> Beowulf::selfdroplast` [protected]

self-message queue drop policy

Definition at line 204 of file Beowulf.H.

Referenced by [exportOptions\(\)](#), and [send\(\)](#).

**10.26.4.4** [ModelParam<int>](#) [Beowulf::selfqlen](#) [protected]

self-message queue length

Definition at line 203 of file Beowulf.H.

Referenced by [exportOptions\(\)](#), and [send\(\)](#).

The documentation for this class was generated from the following files:

- [Beowulf.H](#)
- [Beowulf.C](#)

## 10.27 BestEntry Class Reference

The nearest entry in the tree to the target entry.

```
#include <KDTree.H>
```

### Public Member Functions

- [BestEntry](#) ()  
*Constructor.*
- [BestEntry](#) (const Keypoint &neighbor, int distSq)  
*Constructor.*
- [BestEntry](#) (const Keypoint &neighbor, double dist)  
*Constructor.*
- const double [getDistance](#) () const  
*Get the distance between the neighbor point and the target point.*
- void [setDistance](#) (double value)  
*Set the distance between the neighbor point and the target point.*
- const int [getDistanceSq](#) () const  
*Get the distance squared between the neighbor point and the target point.*
- void [setDistanceSq](#) (int value)  
*Set the distance squared between the neighbor point and the target point.*
- const Keypoint & [getNeighbor](#) () const  
*Get the neighbor point.*
- bool [equals](#) (const [BestEntry](#) &be2) const  
*Returns whether or not this entry is equal to the specified entry.*
- int [compareTo](#) (const [BestEntry](#) &be2) const  
*Compares this entry to the specified entry.*

### 10.27.1 Detailed Description

The nearest entry in the tree to the target entry.

Definition at line 44 of file KDTree.H.

## 10.27.2 Constructor & Destructor Documentation

### 10.27.2.1 BestEntry::BestEntry ()

Constructor.

Definition at line 47 of file KDTree.C.

### 10.27.2.2 BestEntry::BestEntry (const Keypoint & *neighbor*, int *distSq*)

Constructor.

Definition at line 52 of file KDTree.C.

References `sqrt()`.

### 10.27.2.3 BestEntry::BestEntry (const Keypoint & *neighbor*, double *dist*)

Constructor.

Definition at line 60 of file KDTree.C.

## 10.27.3 Member Function Documentation

### 10.27.3.1 int BestEntry::compareTo (const BestEntry & *be2*) const

Compares this entry to the specified entry.

Definition at line 108 of file KDTree.C.

References `getDistanceSq()`.

### 10.27.3.2 bool BestEntry::equals (const BestEntry & *be2*) const

Returns whether or not this entry is equal to the specified entry.

Definition at line 100 of file KDTree.C.

References `getNeighbor()`.

### 10.27.3.3 const double BestEntry::getDistance () const

Get the distance between the neighbor point and the target point.

Definition at line 68 of file KDTree.C.

**10.27.3.4    `const int BestEntry::getDistanceSq () const`**

Get the distance squared between the neighbor point and the target point.

Definition at line 81 of file KDTree.C.

Referenced by `compareTo()`.

**10.27.3.5    `const Keypoint & BestEntry::getNeighbor () const`**

Get the neighbor point.

Definition at line 94 of file KDTree.C.

Referenced by `equals()`.

**10.27.3.6    `void BestEntry::setDistance (double value)`**

Set the distance between the neighbor point and the target point.

Definition at line 74 of file KDTree.C.

**10.27.3.7    `void BestEntry::setDistanceSq (int value)`**

Set the distance squared between the neighbor point and the target point.

Definition at line 87 of file KDTree.C.

References `sqrt()`.

The documentation for this class was generated from the following files:

- [KDTree.H](#)
- [KDTree.C](#)



## 10.28 BitObject Class Reference

Object defined by a connected binary pixel region.

```
#include <BitObject.H>
```

Collaboration diagram for BitObject:

### Public Types

- enum [Coords](#) { **OBJECT**, **IMAGE** }  
*Coordinate system for return values.*

### Public Member Functions

- [BitObject](#) ()  
*default Constructor*
- [BitObject](#) (const [Image](#)< [byte](#) > &img, const [Point2D](#) location, const [byte](#) threshold=1)  
*Constructor from a byte [Image](#) and a location.*
- [BitObject](#) (const [Image](#)< [byte](#) > &img)  
*Construct from a byte [Image](#).*
- [BitObject](#) (std::istream &is)  
*Construct from the data contained in the input stream is.*
- [Image](#)< [byte](#) > [reset](#) (const [Image](#)< [byte](#) > &img, const [Point2D](#) location, const [byte](#) threshold=1)  
*Reset to a new object.*
- int [reset](#) (const [Image](#)< [byte](#) > &img)  
*Reset to a new object.*
- void [freeMem](#) ()  
*delete all stored data, makes the object invalid*
- void [writeToStream](#) (std::ostream &os) const  
*write the entire BitObject to the output stream os*
- void [readFromStream](#) (std::istream &is)

*read the BitObject from the input stream is*

- `template<class T> void setMaxMinAvgIntensity (const Image< T > &img)`  
*Extract the maximum, minimum and average intensity over the object shape in img.*
- `void getMaxMinAvgIntensity (float &maxIntensity, float &minIntensity, float &avgIntensity)`  
*Return the maximum, minimum and average intensity.*
- `Rectangle getBoundingBox (const Coords coords=IMAGE) const`  
*Returns the bounding box of the object.*
- `Image< byte > getObjectMask (const byte value=byte(1), const Coords coords=IMAGE) const`  
*Returns the object as a binary mask.*
- `Dims getObjectDims () const`  
*The dimensions of the bounding box of the object.*
- `Point2D getObjectOrigin () const`  
*The location of the upper left corner of the boundary box in the original image.*
- `Point2D getCentroid (const Coords coords=IMAGE) const`  
*The centroid of the object rounded to integer coordinates.*
- `Vector2D getCentroidXY (const Coords coords=IMAGE) const`  
*The centroid, not rounded.*
- `int getArea () const`  
*The area (number of pixels) of the object.*
- `void getSecondMoments (float &uxxx, float &uyyy, float &uxxy)`  
*Returns the second moments of the object (normalized by the area).*
- `float getMajorAxis ()`  
*Returns the length of the major axis of the object.*
- `float getMinorAxis ()`  
*Returns the length of the minor axis of the object.*
- `float getElongation ()`  
*Returns the elongation of the object (major axis / minor axis).*

- float [getOriAngle](#) ()  
*Returns the angle between the major axis and the x axis.*
- bool [isValid](#) () const  
*whether the object is valid*
- bool [doesIntersect](#) (const [BitObject](#) &other) const  
*returns the intersection of this BitObject with other*
- template<class T\_or\_RGB> void [drawShape](#) ([Image](#)< T\_or\_RGB > &, const T\_or\_RGB &color, float opacity=1.0F)  
*draw the shape of this BitObject into img with color*
- template<class T\_or\_RGB> void [drawOutline](#) ([Image](#)< T\_or\_RGB > &, const T\_or\_RGB &color, float opacity=1.0F)  
*draw the outline of this BitObject into img with color*
- template<class T\_or\_RGB> void [drawBoundingBox](#) ([Image](#)< T\_or\_RGB > &, const T\_or\_RGB &color, float opacity=1.0F)  
*draw the bounding box of this BitObject into img with color*
- template<class T\_or\_RGB> void [draw](#) ([BitObjectDrawMode](#) mode, [Image](#)< T\_or\_RGB > &, const T\_or\_RGB &color, float opacity=1.0F)  
*draw this BitObject according to mode*

### 10.28.1 Detailed Description

Object defined by a connected binary pixel region.

This class extracts a connected binary pixel region from a grayscale image and analyzes a few of its properties

Definition at line 53 of file BitObject.H.

### 10.28.2 Member Enumeration Documentation

#### 10.28.2.1 enum [BitObject::Coords](#)

Coordinate system for return values.

These values are used to specify whether return values should be given in coordinates of the extracted object or in coordinates of the image from which the object was extracted.

Definition at line 107 of file BitObject.H.

## 10.28.3 Constructor & Destructor Documentation

### 10.28.3.1 BitObject::BitObject ()

default Constructor

Definition at line 52 of file BitObject.C.

References freeMem().

### 10.28.3.2 BitObject::BitObject (const Image< byte > & img, const Point2D location, const byte threshold = 1)

Constructor from a byte Image and a location.

**Parameters:**

*img* Image from which the object is extracted

*location* a point that is inside the object - the object is flooded starting from this seed

*threshold* for flooding

Definition at line 58 of file BitObject.C.

References reset().

### 10.28.3.3 BitObject::BitObject (const Image< byte > & img)

Construct from a byte Image.

**Parameters:**

*img* image containing only the object the object pixels are 1, all other pixels are 0

Definition at line 65 of file BitObject.C.

References reset().

### 10.28.3.4 BitObject::BitObject (std::istream & is)

Construct from the data contained in the input stream is.

Definition at line 71 of file BitObject.C.

References readFromStream().

## 10.28.4 Member Function Documentation

### 10.28.4.1 `bool BitObject::doesIntersect (const BitObject & other) const`

returns the intersection of this BitObject with other

if the intersection is empty, an invalid BitObject is returned

Definition at line 526 of file BitObject.C.

References Rectangle::bottom(), byte, crop(), getBoundingBox(), getObjectMask(), isValid(), Rectangle::left(), Rectangle::right(), sum(), takeMin(), and Rectangle::top().

Referenced by VisualEvent::doesIntersect().

### 10.28.4.2 `template<class T_or_RGB> void BitObject::draw (BitObjectDrawMode mode, Image< T_or_RGB > &, const T_or_RGB & color, float opacity = 1.0F)`

draw this BitObject according to mode

Definition at line 687 of file BitObject.C.

References BODMbbox, BODMoutline, BODMshape, drawBoundingBox(), drawOutline(), and drawShape().

Referenced by VisualEventSet::drawTokens().

### 10.28.4.3 `template<class T_or_RGB> void BitObject::drawBoundingBox (Image< T_or_RGB > &, const T_or_RGB & color, float opacity = 1.0F)`

draw the bounding box of this BitObject into img with color

Definition at line 657 of file BitObject.C.

References ASSERT, Rectangle::bottom(), Image< T >::getDims(), Image< T >::getVal(), Image< T >::initialized(), isValid(), Rectangle::left(), Rectangle::right(), Image< T >::setVal(), and Rectangle::top().

Referenced by draw().

### 10.28.4.4 `template<class T_or_RGB> void BitObject::drawOutline (Image< T_or_RGB > &, const T_or_RGB & color, float opacity = 1.0F)`

draw the outline of this BitObject into img with color

Definition at line 613 of file BitObject.C.

References ASSERT, Rectangle::bottom(), byte, Image< byte >::coordsOk(), Image< T >::coordsOk(), Image< T >::getDims(), Image< T >::getVal(), Image< byte >::getVal(), Image< T >::initialized(), isValid(), Rectangle::left(), Rectangle::right(), Image< T >::setVal(), Rectangle::top(), and ZEROS.

Referenced by draw().

#### 10.28.4.5 **template<class T\_or\_RGB> void BitObject::drawShape (Image< T\_or\_RGB > &, const T\_or\_RGB & color, float opacity = 1.0F)**

draw the shape of this BitObject into img with color

Definition at line 585 of file BitObject.C.

References ASSERT, Image< byte >::begin(), Image< T >::beginw(), Rectangle::bottom(), Image< T >::getDims(), Image< T >::getWidth(), Image< T >::initialized(), isValid(), Rectangle::left(), Rectangle::right(), and Rectangle::top().

Referenced by draw().

#### 10.28.4.6 **void BitObject::freeMem ()**

delete all stored data, makes the object invalid

Definition at line 249 of file BitObject.C.

References Image< byte >::freeMem().

Referenced by BitObject(), and reset().

#### 10.28.4.7 **int BitObject::getArea () const**

The area (number of pixels) of the object.

Definition at line 484 of file BitObject.C.

Referenced by VisualEvent::assign(), VisualEventSet::drawTokens(), extractBitObjects(), VisualEvent::getPropertyVector(), and VisualEvent::VisualEvent().

#### 10.28.4.8 **Rectangle BitObject::getBoundingBox (const Coords coords = IMAGE) const**

Returns the bounding box of the object.

Definition at line 416 of file BitObject.C.

References Rectangle::height(), and Rectangle::width().

Referenced by `doesIntersect()`, `VisualEventSet::drawTokens()`, and `MbariResultViewer::saveSingleEventFrame()`.

#### 10.28.4.9 **Point2D** BitObject::getCentroid (const **Coords** coords = IMAGE) const

The centroid of the object rounded to integer coordinates.

Definition at line 465 of file `BitObject.C`.

References `getCentroidXY()`, and `Vector2D::getPoint2D()`.

#### 10.28.4.10 **Vector2D** BitObject::getCentroidXY (const **Coords** coords = IMAGE) const

The centroid, not rounded.

Definition at line 471 of file `BitObject.C`.

References `getObjectOrigin()`.

Referenced by `getCentroid()`.

#### 10.28.4.11 float BitObject::getElongation ()

Returns the elongation of the object (major axis / minor axis).

Definition at line 509 of file `BitObject.C`.

Referenced by `VisualEvent::getPropertyVector()`.

#### 10.28.4.12 float BitObject::getMajorAxis ()

Returns the length of the major axis of the object.

Definition at line 495 of file `BitObject.C`.

Referenced by `VisualEvent::getPropertyVector()`.

#### 10.28.4.13 void BitObject::getMaxMinAvgIntensity (float & maxIntensity, float & minIntensity, float & avgIntensity)

Return the maximum, minimum and average intensity.

See `setMinMaxAvgIntensity` for details

Definition at line 406 of file `BitObject.C`.

Referenced by `VisualEvent::getPropertyVector()`.

**10.28.4.14 float BitObject::getMinorAxis ()**

Returns the length of the minor axis of the object.

Definition at line 502 of file BitObject.C.

Referenced by VisualEvent::getPropertyVector().

**10.28.4.15 Dims BitObject::getObjectDims () const**

The dimensions of the bounding box of the object.

Definition at line 455 of file BitObject.C.

References Image< byte >::getDims().

**10.28.4.16 Image< byte > BitObject::getObjectMask (const byte value = byte(1), const Coords coords = IMAGE) const**

Returns the object as a binary mask.

**Parameters:**

*value* the value that is used for the object

Definition at line 430 of file BitObject.C.

References ASSERT, byte, getObjectOrigin(), isValid(), pasteImage(), replaceVals(), and ZEROS.

Referenced by doesIntersect().

**10.28.4.17 Point2D BitObject::getObjectOrigin () const**

The location of the upper left corner of the boundary box in the original image.

Definition at line 459 of file BitObject.C.

References Rectangle::left(), and Rectangle::top().

Referenced by getCentroidXY(), and getObjectMask().

**10.28.4.18 float BitObject::getOriAngle ()**

Returns the angle between the major axis and the x axis.

Definition at line 516 of file BitObject.C.

Referenced by VisualEvent::getPropertyVector().



**10.28.4.19 void BitObject::getSecondMoments (float & uxx, float & uyy, float & uxy)**

Returns the second moments of the object (normalized by the area).

Definition at line 488 of file BitObject.C.

Referenced by VisualEvent::getPropertyVector().

**10.28.4.20 bool BitObject::isValid () const**

whether the object is valid

This is going to be false if no object could be extracted from the image that was given at the constructor or at reset

Definition at line 522 of file BitObject.C.

References Rectangle::isValid().

Referenced by doesIntersect(), drawBoundingBox(), drawOutline(), drawShape(), VisualEventSet::drawTokens(), getObjectMask(), VisualEvent::getPropertyVector(), and setMaxMinAvgIntensity().

**10.28.4.21 void BitObject::readFromStream (std::istream & is)**

read the BitObject from the input stream is

Definition at line 321 of file BitObject.C.

References readBinImage().

Referenced by BitObject().

**10.28.4.22 int BitObject::reset (const Image< byte > & img)**

Reset to a new object.

**Parameters:**

*img* image containing only the object the object pixels are 1, all other pixels are 0;

**Returns:**

the area of the extracted object; -1 if no object could be extracted - in this case the BitObject is invalid

Definition at line 127 of file BitObject.C.

References ASSERT, crop(), freeMem(), getCentroidFirstLast(), Image< T >::getDims(), Image< T >::initialized(), Rectangle::modify(), Vector2D::reset(), and sumXY().

#### 10.28.4.23 **Image< byte > BitObject::reset (const Image< byte > & img, const Point2D location, const byte threshold = 1)**

Reset to a new object.

##### **Parameters:**

**img** Image from which the object is extracted

**location** a point that is inside the object - the object is flooded starting from this seed

**threshold** for flooding

##### **Returns:**

a mask of the extracted object (in IMAGE coordinates)

Definition at line 77 of file BitObject.C.

References ASSERT, byte, crop(), floodCleanBB(), freeMem(), getCentroidFirstLast(), Image< T >::getDims(), Image< byte >::getHeight(), Image< byte >::getWidth(), Image< T >::initialized(), Rectangle::left(), Vector2D::reset(), sumXY(), and Rectangle::top().

Referenced by BitObject(), and extractBitObjects().

#### 10.28.4.24 **template<class T> void BitObject::setMaxMinAvgIntensity (const Image< T > & img)**

Extract the maximum, minimum and average intensity over the object shape in img.

The results are stored in BitObject and can be accessed using getMaxMinAvgIntensity

Definition at line 365 of file BitObject.C.

References ASSERT, Image< T >::begin(), Image< byte >::begin(), Rectangle::bottom(), byte, Image< T >::getDims(), Image< T >::getWidth(), isValid(), Rectangle::left(), Rectangle::right(), and Rectangle::top().

#### 10.28.4.25 **void BitObject::writeToStream (std::ostream & os) const**

write the entire BitObject to the output stream os

Definition at line 269 of file BitObject.C.

References `Rectangle::bottom()`, `Dims::h()`, `Rectangle::isValid()`, `Rectangle::left()`, `Rectangle::right()`, `Rectangle::top()`, `Dims::w()`, `writeBinImage()`, and `Vector2D::writeToStream()`.

Referenced by `Token::writeToStream()`.

The documentation for this class was generated from the following files:

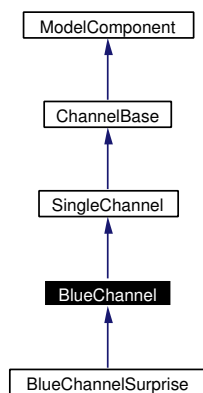
- [BitObject.H](#)
- [BitObject.C](#)

## 10.29 BlueChannel Class Reference

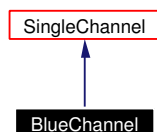
A blue color channel.

```
#include <Channels.H>
```

Inheritance diagram for BlueChannel:



Collaboration diagram for BlueChannel:



### Public Member Functions

- [BlueChannel](#) ([ModelManager](#) &mgr)  
*Construct with standard params.*
- virtual [~BlueChannel](#) ()  
*Destructor.*

### 10.29.1 Detailed Description

A blue color channel.

Definition at line 207 of file Channels.H.

## 10.29.2 Constructor & Destructor Documentation

### 10.29.2.1 BlueChannel::BlueChannel ([ModelManager](#) & *mgr*)

Construct with standard params.

Definition at line 253 of file Channels.C.

References [B](#), [makeShared\(\)](#), and [ModelParam< bool >::setVal\(\)](#).

### 10.29.2.2 BlueChannel::~BlueChannel () [virtual]

Destructor.

Definition at line 260 of file Channels.C.

The documentation for this class was generated from the following files:

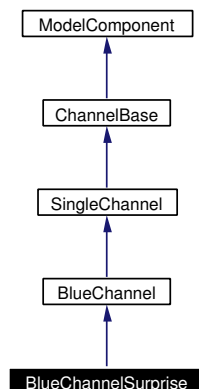
- [Channels.H](#)
- [Channels.C](#)

## 10.30 BlueChannelSurprise Class Reference

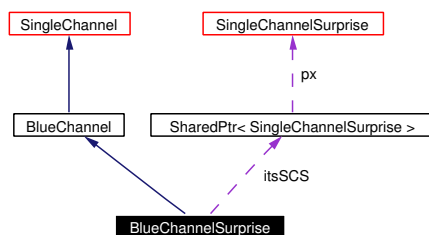
A blue color channel.

```
#include <ChannelsSurprise.H>
```

Inheritance diagram for BlueChannelSurprise:



Collaboration diagram for BlueChannelSurprise:



### Public Member Functions

- `BlueChannelSurprise (ModelManager &mgr)`  
*Construct with standard params.*
- `virtual ~BlueChannelSurprise ()`  
*Destructor.*
- `virtual Image< float > computeSubmap (const uint index) const`  
*`computeSubmap()` is special and will return surprise*

## Protected Attributes

- [SharedPtr](#)< [SingleChannelSurprise](#) > *itsSCS*  
*our surprise computer*

### 10.30.1 Detailed Description

A blue color channel.

Definition at line 231 of file ChannelsSurprise.H.

### 10.30.2 Constructor & Destructor Documentation

#### 10.30.2.1 BlueChannelSurprise::BlueChannelSurprise ([ModelManager](#) & *mgr*)

Construct with standard params.

Definition at line 214 of file ChannelsSurprise.C.

References [ModelComponent::addSubComponent\(\)](#).

#### 10.30.2.2 BlueChannelSurprise::~~BlueChannelSurprise () [virtual]

Destructor.

Definition at line 220 of file ChannelsSurprise.C.

### 10.30.3 Member Function Documentation

#### 10.30.3.1 [Image](#)< float > BlueChannelSurprise::computeSubmap (const [uint index](#)) const [virtual]

[computeSubmap\(\)](#) is special and will return surprise

Reimplemented from [SingleChannel](#).

Definition at line 223 of file ChannelsSurprise.C.

References [uint](#).

## 10.30.4 Member Data Documentation

### 10.30.4.1 [SharedPtr<SingleChannelSurprise> BlueChannelSurprise::itsSCS](#) [protected]

our surprise computer

Definition at line 244 of file ChannelsSurprise.H.

The documentation for this class was generated from the following files:

- [ChannelsSurprise.H](#)
- ChannelsSurprise.C

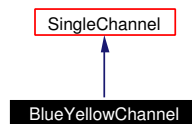


## 10.31 BlueYellowChannel Class Reference

A blue-yellow double-opponent color channel.

```
#include <Channels.H>
```

Inheritance diagram for BlueYellowChannel: Collaboration diagram for BlueYellowChannel:



### Public Member Functions

- [BlueYellowChannel](#) ([ModelManager](#) &mgr)  
*Construct with standard params.*
- virtual [~BlueYellowChannel](#) ()  
*Destructor.*

#### 10.31.1 Detailed Description

A blue-yellow double-opponent color channel.

Definition at line 125 of file Channels.H.

### 10.31.2 Constructor & Destructor Documentation

#### 10.31.2.1 BlueYellowChannel::BlueYellowChannel ([ModelManager](#) & mgr)

Construct with standard params.

Definition at line 165 of file Channels.C.

References BY, makeShared(), and ModelParam< bool >::setVal().

#### 10.31.2.2 BlueYellowChannel::~~BlueYellowChannel () [virtual]

Destructor.

Definition at line 172 of file Channels.C.

The documentation for this class was generated from the following files:

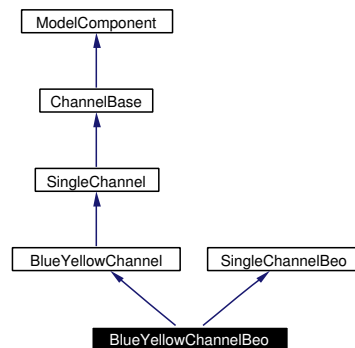
- [Channels.H](#)
- [Channels.C](#)

## 10.32 BlueYellowChannelBeo Class Reference

Beowulf version of [BlueYellowChannel](#).

```
#include <ChannelsBeo.H>
```

Inheritance diagram for BlueYellowChannelBeo:



Collaboration diagram for BlueYellowChannelBeo:

### Public Member Functions

- **BlueYellowChannelBeo** ([ModelManager](#) &mgr, const [SharedPtr](#)< [Beowulf](#) > &beow)
- virtual bool [outputAvailable](#) () const  
*Query whether the channel is ready to give yield valid output.*

### Protected Member Functions

- virtual void [doInput](#) (const [Image](#)< [PixRGB](#)< float > > \*, const [Image](#)< float > \*bwimg, const double t, const [Image](#)< byte > &clipMask)  
*[SingleChannel](#) requires only luminance input.*
- virtual void [start1](#) ()

#### 10.32.1 Detailed Description

Beowulf version of [BlueYellowChannel](#).

Definition at line 82 of file ChannelsBeo.H.

## 10.32.2 Member Function Documentation

**10.32.2.1 void BlueYellowChannelBeo::doInput (const [Image](#)< [PixRGB](#)< float > > \*, const [Image](#)< float > \* *bwimg*, const double *t*, const [Image](#)< [byte](#) > & *clipMask*)** [protected, virtual]

[SingleChannel](#) requires only luminance input.

Reimplemented from [SingleChannel](#).

Definition at line 109 of file ChannelsBeo.C.

References [SingleChannelBeo::sendInput\(\)](#).

**10.32.2.2 bool BlueYellowChannelBeo::outputAvailable () const** [virtual]

Query whether the channel is ready to give yield valid output.

In some cases (e.g. with motion channels), it may take several input frames before the channel is fully initialized.

Reimplemented from [SingleChannel](#).

Definition at line 102 of file ChannelsBeo.C.

References [SingleChannel::outputAvailable\(\)](#).

**10.32.2.3 void BlueYellowChannelBeo::start1 ()** [protected, virtual]

If you overload this, make sure you call [SingleChannel::start1\(\)](#) at the beginning of your overload

Reimplemented from [SingleChannel](#).

Definition at line 113 of file ChannelsBeo.C.

References [SingleChannelBeo::requestBeowulfNode\(\)](#), and [SingleChannel::start1\(\)](#).

The documentation for this class was generated from the following files:

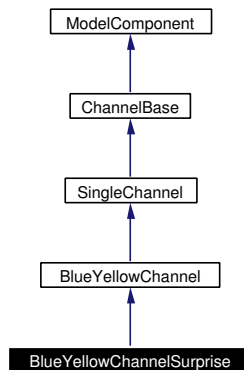
- [ChannelsBeo.H](#)
- [ChannelsBeo.C](#)

## 10.33 BlueYellowChannelSurprise Class Reference

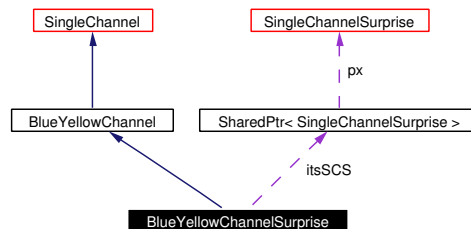
A blue-yellow double-opponent color channel.

```
#include <ChannelsSurprise.H>
```

Inheritance diagram for BlueYellowChannelSurprise:



Collaboration diagram for BlueYellowChannelSurprise:



### Public Member Functions

- [BlueYellowChannelSurprise](#) ([ModelManager](#) &mgr)  
*Construct with standard params.*
- virtual [~BlueYellowChannelSurprise](#) ()  
*Destructor.*
- virtual [Image< float > computeSubmap](#) (const [uint](#) index) const  
*[computeSubmap\(\)](#) is special and will return surprise*

## Protected Attributes

- [SharedPtr< SingleChannelSurprise > itsSCS](#)  
*our surprise computer*

### 10.33.1 Detailed Description

A blue-yellow double-opponent color channel.

Definition at line 105 of file ChannelsSurprise.H.

### 10.33.2 Constructor & Destructor Documentation

#### 10.33.2.1 [BlueYellowChannelSurprise::BlueYellowChannelSurprise](#) ([ModelManager](#) & *mgr*)

Construct with standard params.

Definition at line 109 of file ChannelsSurprise.C.

References [ModelComponent::addSubComponent\(\)](#).

#### 10.33.2.2 [BlueYellowChannelSurprise::~~BlueYellowChannelSurprise](#) () [virtual]

Destructor.

Definition at line 115 of file ChannelsSurprise.C.

### 10.33.3 Member Function Documentation

#### 10.33.3.1 [Image< float > BlueYellowChannelSurprise::computeSubmap](#) (const [uint index](#)) const [virtual]

[computeSubmap\(\)](#) is special and will return surprise

Reimplemented from [SingleChannel](#).

Definition at line 118 of file ChannelsSurprise.C.

References [uint](#).

## 10.33.4 Member Data Documentation

### 10.33.4.1 [SharedPtr<SingleChannelSurprise> BlueYellowChannelSurprise::itsSCS](#) [protected]

our surprise computer

Definition at line 118 of file ChannelsSurprise.H.

The documentation for this class was generated from the following files:

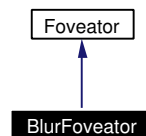
- [ChannelsSurprise.H](#)
- ChannelsSurprise.C

## 10.34 BlurFoveator Class Reference

A class that does space-variant processing with progressive blurring.

```
#include <BlurFoveator.H>
```

Inheritance diagram for BlurFoveator:



Collaboration diagram for BlurFoveator:

### Public Member Functions

- [BlurFoveator](#) (const [Image](#)< [PixRGB](#)< [byte](#) > > &img, int filterSize)  
*Construct a BlurFoveator for an image.*
- [BlurFoveator](#) (const [BlurFoveator](#) &bf)  
*Construct a BlurFoveator from another.*
- [BlurFoveator](#) & [operator=](#) (const [BlurFoveator](#) &bf)  
*Construct a BlurFoveator equal to another.*
- [~BlurFoveator](#) ()  
*Destructor.*
- [Image](#)< [PixRGB](#)< [byte](#) > > [foveate](#) (void)  
*Perform blur foveation and return resulting image.*

### Static Public Member Functions

- [Image](#)< [PixRGB](#)< [byte](#) > > [foveate](#) (const [Image](#)< [PixRGB](#)< [byte](#) > > &img, int filterSize, int x, int y)  
*Perform one blur foveation without a BlurFoveator.*



### 10.34.1 Detailed Description

A class that does space-variant processing with progressive blurring.

A Gaussian blur is applied throughout the image, with small blur radius near the foveation origin, which increases as we go further into the periphery. This causes the image to appear sharp near the origin while blurry toward the outer edges.

Testing has revealed that BlurFoveator is not as efficient as [PyrFoveator](#) or [LPT-Foveator](#) but can still achieve the desired result.

Definition at line 18 of file BlurFoveator.H.

### 10.34.2 Constructor & Destructor Documentation

#### 10.34.2.1 `BlurFoveator::BlurFoveator (const Image< PixRGB< byte > > & img, int filterSize)`

Construct a BlurFoveator for an image.

This constructor will create a BlurFoveator that handles *img* and all other images of the same dimensions.

**Parameters:**

*img* image to be foveated

*filterSize* size of blur filter to be applied

Definition at line 9 of file BlurFoveator.C.

Referenced by operator=().

#### 10.34.2.2 `BlurFoveator::BlurFoveator (const BlurFoveator & bf)`

Construct a BlurFoveator from another.

Definition at line 35 of file BlurFoveator.C.

References *gbms*, and *matrices*.

#### 10.34.2.3 `BlurFoveator::~~BlurFoveator ()`

Destructor.

Definition at line 55 of file BlurFoveator.C.

### 10.34.3 Member Function Documentation

#### 10.34.3.1 `Image< PixRGB< byte > > BlurFoveator::foveate (const Image< PixRGB< byte > > & img, int filterSize, int x, int y) [static]`

Perform one blur foveation without a BlurFoveator.

This function is for foveating ONE image with a call such as `BlurFoveator::foveate()`. It is not optimized for foveating multiple images.

**Parameters:**

*img* image to be foveated  
*filterSize* size of blur filter to be applied  
*x* horizontal coordinate of origin  
*y* vertical coordinate of origin

Definition at line 314 of file BlurFoveator.C.

References `Image< T >::coordsOk()`, `NO_INIT`, and `Image< T >::setVal()`.

#### 10.34.3.2 `Image< PixRGB< byte > > BlurFoveator::foveate (void) [virtual]`

Perform blur foveation and return resulting image.

Implements `Foveator`.

Definition at line 226 of file BlurFoveator.C.

References `Image< T >::coordsOk()`, `Image< PixRGB< byte > >::getVal()`, `Point2D::i`, `Point2D::j`, `NO_INIT`, and `Image< T >::setVal()`.

Referenced by `main()`.

#### 10.34.3.3 `BlurFoveator & BlurFoveator::operator= (const BlurFoveator & bf)`

Construct a BlurFoveator equal to another.

Definition at line 48 of file BlurFoveator.C.

References `BlurFoveator()`.

The documentation for this class was generated from the following files:

- `BlurFoveator.H`
- `BlurFoveator.C`

## 10.35 BPneuron Class Reference

neuron for a back prop neural net ([BPnnet](#))

```
#include <BPneuron.H>
```

### Public Member Functions

- [BPneuron \(\)](#)  
*Constructor.*
- [~BPneuron \(\)](#)  
*Destructor.*
- void [assignInput](#) (double input)  
*Assign given weighted input sum [S\_j] and activation level.*
- double [calcOutputDelta](#) (const double target)  
*Calculate and set delta for output neurons.*
- void [calcHiddenDelta](#) (double weightedDeltaSum)  
*Calculate and set delta for hidden neurons.*
- double [getDelta](#) ()  
*Return delta (error level).*
- double [getActivationLevel](#) ()  
*Return activation level.*

### 10.35.1 Detailed Description

neuron for a back prop neural net ([BPnnet](#))

Definition at line 41 of file BPneuron.H.

### 10.35.2 Constructor & Destructor Documentation

#### 10.35.2.1 BPneuron::BPneuron ()

Constructor.

Definition at line 41 of file BPneuron.C.

#### 10.35.2.2 BPneuron::~~BPneuron ()

Destructor.

Definition at line 49 of file BPneuron.C.

### 10.35.3 Member Function Documentation

#### 10.35.3.1 void BPneuron::assignInput (double *input*)

Assign given weighted input sum [S<sub>j</sub>] and activation level.

Activation Level  $a_j = f(S_j)$

Definition at line 53 of file BPneuron.C.

#### 10.35.3.2 void BPneuron::calcHiddenDelta (double *weightedDeltaSum*)

Calculate and set delta for hidden neurons.

For hidden layer neurons; compute hidden delta from weighted sum of output deltas [delta<sub>j</sub> = sum<sub>k</sub>(delta<sub>k</sub> \* w<sub>kj</sub>)\*f'(S<sub>j</sub>)] returns delta<sub>j</sub>

Definition at line 69 of file BPneuron.C.

#### 10.35.3.3 double BPneuron::calcOutputDelta (const double *target*)

Calculate and set delta for output neurons.

For output layer neurons; finds difference between output and target and multiplies by derivative of sigmoid function [delta<sub>j</sub> = (t<sub>j</sub> - a<sub>j</sub>)\*f'(S<sub>j</sub>)] returns the difference [t<sub>j</sub> - a<sub>j</sub>] for RMS error calculation

Definition at line 60 of file BPneuron.C.

#### 10.35.3.4 double BPneuron::getActivationLevel ()

Return activation level.

Definition at line 82 of file BPneuron.C.

#### 10.35.3.5 double BPneuron::getDelta ()

Return delta (error level).

Definition at line 76 of file BPneuron.C.

The documentation for this class was generated from the following files:

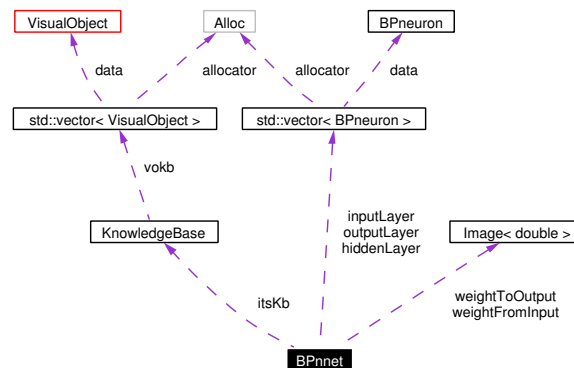
- [BPneuron.H](#)
- [BPneuron.C](#)

## 10.36 BPnnnet Class Reference

describes structure of a 3 layer back prop neural net

```
#include <BPnnnet.H>
```

Collaboration diagram for BPnnnet:



### Public Member Functions

- **BPnnnet** (const int numInput, const int numHidden, const **KnowledgeBase** \*kb)  
*Constructor.*
- **~BPnnnet** ()  
*Destructor.*
- void **randomizeWeights** ()  
*Initialize all weights to (small) random values.*
- void **normalizeWeights** ()  
*Normalize weights.*
- double **train** (const **Image**< float > &in, const **VisualObject** &target, const double learnRate)  
*Do one training iteration.*
- bool **recognize** (const **Image**< float > &in, **VisualObject** &vo)  
*Attempt to recognize given jet as a certain visual object.*
- bool **save** (const char \*filename) const

*Store 2 matrices of weights to the file "filename".*

- bool **load** (const char \***filename**)

*Assign all weights based on data stored in the file "filename".*

### 10.36.1 Detailed Description

describes structure of a 3 layer back prop neural net

Definition at line 47 of file BPnnnet.H.

### 10.36.2 Constructor & Destructor Documentation

#### 10.36.2.1 BPnnnet::BPnnnet (const int *numInput*, const int *numHidden*, const KnowledgeBase \* *kb*)

Constructor.

Definition at line 46 of file BPnnnet.C.

References KnowledgeBase::getSize(), and Image< double >::resize().

#### 10.36.2.2 BPnnnet::~BPnnnet ()

Destructor.

Definition at line 73 of file BPnnnet.C.

References Image< double >::freeMem().

### 10.36.3 Member Function Documentation

#### 10.36.3.1 bool BPnnnet::load (const char \* *filename*)

*Assign all weights based on data stored in the file "filename".*

returns false if the # of units in each layer do not match the matrix sizes in the file

Definition at line 247 of file BPnnnet.C.

References Image< double >::getArrayPtr(), and Image< double >::getSize().

#### 10.36.3.2 void BPnnnet::normalizeWeights ()

Normalize weights.

Definition at line 94 of file BPnnnet.C.

References `Image< double >::clamp()`, and `Image< double >::getMinMax()`.

Referenced by `randomizeWeights()`.

#### 10.36.3.3 `void BPnnnet::randomizeWeights ()`

Initialize all weights to (small) random values.

Definition at line 80 of file BPnnnet.C.

References `normalizeWeights()`, and `Image< double >::setVal()`.

#### 10.36.3.4 `bool BPnnnet::recognize (const Image< float > & in, VisualObject & vo)`

Attempt to recognize given jet as a certain visual object.

Definition at line 191 of file BPnnnet.C.

References `VisualObject::getName()`, and `KnowledgeBase::getVisualObject()`.

#### 10.36.3.5 `bool BPnnnet::save (const char * filename) const`

Store 2 matrices of weights to the file "filename".

use this after training a net

Definition at line 227 of file BPnnnet.C.

References `Image< double >::getArrayPtr()`, and `Image< double >::getSize()`.

#### 10.36.3.6 `double BPnnnet::train (const Image< float > & in, const VisualObject & target, const double learnRate)`

Do one training iteration.

returns  $rms1 = \text{sum of } (\text{target output} - \text{actual output})^2 / n_{\text{outputs}}$  for all output layer neurons

Definition at line 106 of file BPnnnet.C.

References `KnowledgeBase::findVisualObjectIndex()`, `Image< double >::getVal()`, and `Image< double >::setVal()`.

The documentation for this class was generated from the following files:

- [BPnnnet.H](#)
- [BPnnnet.C](#)

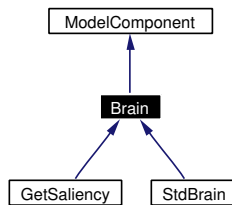


## 10.37 Brain Class Reference

This is the brain, a holder for various vision objects.

```
#include <Brain.H>
```

Inheritance diagram for Brain:



Collaboration diagram for Brain:

### Public Member Functions

- `const Dims & getInputDims () const`  
*Get input dimensions.*
- `Image< PixRGB< byte > > getRawInput () const`  
*Get whatever raw input we have received before passing it through [Retina](#).*
- `Image< byte > getObjectMask () const`  
*Returns binary mask of currently attended object.*
- `Image< byte > getVisualField () const`  
*Returns current visual field.*
- `void findFixation (Point2D &fixation, float &value) const`  
*Finds the current fixation in AGM.*
- `void saveResults (const SharedPtr< OutputFrameSeries > &ofs)`  
*Save our internals and outputs from our modules.*

### Constructors, destructors, and run-time "modules"

Typically, you instantiate a [Brain](#) and then add various plug-in "modules" to it by using the `addModule()` function. If you decide so, each module you add may

become a subcomponent of your [Brain](#) object; if so, you should not do a [ModelManager::registerComponent\(\)](#) on them, but just do it once for all on your [Brain](#) object instead. Modules are a restricted set of [ModelComponent](#) objects, see [addModule\(\)](#) for details of which ones are supported.

- [Brain](#) ([ModelManager](#) &mgr, const [std::string](#) &descrName="Brain", const [std::string](#) &tagName="Brain")  
*Constructor.*
- virtual [~Brain](#) ()  
*Virtual destructor for safe inheritance.*
- virtual void [exportOptions](#) (const int whatToExport, const bool recurse=true)  
*Export some of our [ModelParam](#) parameters as command-line-options.*
- void [addModule](#) ([SharedPtr](#)< [ModelComponent](#) > mod, const char \*modtype, const bool makeSubComp=false)  
*Add a module.*
- void [removeModule](#) (const char \*modtype)  
*Remove a previously added module.*
- virtual void [reset](#) (const bool recurse=true)  
*Reset Brain and propagate the reset signal to the sub-components.*
- [SharedPtr](#)< [ModelComponent](#) > [module](#) (const char \*modtype) const  
*Get a module; see shorthands below.*
- [SharedPtr](#)< [Retina](#) > [getRET](#) () const  
*Get the [Retina](#).*
- [SharedPtr](#)< [VisualCortex](#) > [getVC](#) () const  
*Get the [VisualCortex](#).*
- [SharedPtr](#)< [SaccadeController](#) > [getSC](#) () const  
*Get the [SaccadeController](#).*
- [SharedPtr](#)< [SimulationViewer](#) > [getSV](#) () const  
*Get the [SimulationViewer](#).*
- [SharedPtr](#)< [ShapeEstimator](#) > [getSE](#) () const  
*Get the [ShapeEstimator](#).*
- [SharedPtr](#)< [SaliencyMap](#) > [getSM](#) () const  
*Get the [SaliencyMap](#).*

- `SharedPtr< TaskRelevanceMap > getTRM () const`  
*Get the [TaskRelevanceMap](#).*
- `SharedPtr< AttentionGuidanceMap > getAGM () const`  
*Get the [AttentionGuidanceMap](#).*
- `SharedPtr< WinnerTakeAll > getWTA () const`  
*Get the [WinnerTakeAll](#).*
- `SharedPtr< IneroTemporal > getIT () const`  
*Get the [IneroTemporal](#).*
- `SharedPtr< GistEstimator > getGE () const`  
*Get the [GistEstimator](#).*
- `bool hasModule (const char *modtype) const`  
*Returns true if the corresponding module has been added with [addModule\(\)](#).*

#### Brain simulation functions

- `void input (const Image< PixRGB< byte > > &inp, const Image< byte > &targets, const Image< byte > &clipmask=Image< byte >())`  
*Load a new frame and pass it through visual cortex.*
- `void processInput (const Image< PixRGB< byte > > &inp, const Image< byte > &targets, const Image< byte > &clipmask=Image< byte >())`  
*Deliver output to VC & Co (called from input, see there for details).*
- `void obtainOutput ()`  
*Get the output from [VisualCortex](#) & Co. (called from input).*
- `void evolve (double &stime, Point2D &winner, int &ntarg, float &winV)`  
*Evolve saliency map and winner-take-all.*
- `double resetTime ()`  
*Reset the internal time.*
- `void saccadicSuppression (const bool on)`  
*Turn saccadic suppression on/off.*
- `void blinkSuppression (const bool on)`  
*Turn blink suppression on/off.*

## Protected Member Functions

- virtual void [start1](#) ()  
*do some additional config at start time; see [ModelComponent.H](#)*

## Protected Attributes

- [ModelParam](#)< [IORtype](#) > [itsIORtype](#)  
*type of IOR to use. See [IORtypes.H](#)*
- [ModelParam](#)< bool > [itsUseRandom](#)  
*add jitter to winning location?*
- [ModelParam](#)< int > [itsFOAradius](#)  
*radius of the FOA, in pixels*
- [ModelParam](#)< int > [itsFoveaRadius](#)  
*radius of the Fovea, in pixels*
- [ModelParam](#)< double > [itsTimeStep](#)  
*Simulation time step, in seconds.*
- [ModelParam](#)< [LevelSpec](#) > [itsLevelSpec](#)  
*Specification of our image processing center-surround levels.*
- [ModelParam](#)< double > [itsBoringDelay](#)  
*Inter-covert-shift delay (in ms) that makes a shift of attention boring.*
- [ModelParam](#)< float > [itsBoringSMmv](#)
- [ModelParam](#)< float > [itsMaxWinMv](#)
- [ModelParam](#)< bool > [itsBlankBlink](#)  
*Blank out visual inputs during blinks?*
- [ModelParam](#)< bool > [itsSaveObjMask](#)  
*Save our internals when [saveResults\(\)](#) is called?*
- [ModelParam](#)< int > [itsTooManyShifts](#)  
*Exit after a given number of attention shifts (if not zero).*
- [ModelParam](#)< float > [itsTooMuchTime](#)  
*Exit after a given time in seconds (if not zero).*

- [ModelParam](#)< bool > [itsSaveWinnerFeatures](#)  
*Save vector of features at each attended location?*
- [SharedPtr](#)< [Retina](#) > [itsRET](#)  
*Retina module.*
- [SharedPtr](#)< [VisualCortex](#) > [itsVC](#)  
*VisualCortex module.*
- [SharedPtr](#)< [SaccadeController](#) > [itsSC](#)  
*SaccadeController module.*
- [SharedPtr](#)< [SimulationViewer](#) > [itsSV](#)  
*SimulationViewer module.*
- [SharedPtr](#)< [ShapeEstimator](#) > [itsSE](#)  
*ShapeEstimator Module.*
- [SharedPtr](#)< [SaliencyMap](#) > [itsSM](#)  
*SaliencyMap Module.*
- [SharedPtr](#)< [TaskRelevanceMap](#) > [itsTRM](#)  
*TaskRelevanceMap Module.*
- [SharedPtr](#)< [AttentionGuidanceMap](#) > [itsAGM](#)  
*AttentionGuidanceMap Module.*
- [SharedPtr](#)< [WinnerTakeAll](#) > [itsWTA](#)  
*WinnerTakeAll Module.*
- [SharedPtr](#)< [InferoTemporal](#) > [itsIT](#)  
*InferoTemporal Module.*
- [SharedPtr](#)< [GistEstimator](#) > [itsGE](#)  
*GistEstimator Module.*

### 10.37.1 Detailed Description

This is the brain, a holder for various vision objects.

It contains a retina (input image), a [VisualCortex](#), a [SaliencyMap](#), a [WinnerTakeAll](#), a visual field (image that keeps track of attended locations), and lots of interconnections between those.

Definition at line 66 of file Brain.H.

## 10.37.2 Constructor & Destructor Documentation

### 10.37.2.1 `Brain::Brain (ModelManager & mgr, const std::string & descrName = "Brain", const std::string & tagName = "Brain")`

Constructor.

**Parameters:**

*mgr* our [ModelManager](#) (see [ModelManager.H](#))  
*descrName* descriptive name for human useag  
*tagName* name for [ParamMap](#) usage

Definition at line 68 of file Brain.C.

References [IORauto](#), and [IORtype](#).

### 10.37.2.2 `Brain::~Brain () [virtual]`

Virtual destructor for safe inheritance.

Definition at line 99 of file Brain.C.

## 10.37.3 Member Function Documentation

### 10.37.3.1 `void Brain::addModule (SharedPtr< ModelComponent > mod, const char * modtype, const bool makeSubComp = false)`

Add a module.

Supported modules are, with their modtype names:

- "Retina" - derived from class [Retina](#)
- "VisualCortex" - derived from class [VisualCortex](#)
- "SaccadeController" - derived from class [SaccadeController](#)
- "SimulationViewer" - derived from class [SimulationViewer](#)
- "ShapeEstimator" - derived from class [ShapeEstimator](#)

- "SaliencyMap" - derived from class [SaliencyMap](#)
- "TaskRelevanceMap" - derived from class [TaskRelevanceMap](#)
- "AttentionGuidanceMap" - derived from class [AttentionGuidanceMap](#)
- "WinnerTakeAll" - derived from class [WinnerTakeAll](#)
- "InferoTemporal" - derived from class [InferoTemporal](#)
- "GistEstimator" - derived from class [GistEstimator](#)

**Parameters:**

**mod** a [SharedPtr](#) to a [ModelComponent](#) derivative object. For example, to add a module derived from [VisualCortex](#), a typical call would be: `brain.addModule(makeShared(new VCderived(...)), "VisualCortex");`

**modtype** one of the module type names specified above. Anything else will throw a fatal error. Some type checking will be done to enforce that the passed [ModelComponent](#) indeed derives from the expected base class (e.g., if you add a module with modtype "VisualCortex" then we will check that your mod derives from [VisualCortex](#). A fatal error will be thrown if that test fails.

**makeSubComp** If true, the object passed as argument will become a subcomponent of us, thus you will not need to separately register it with [ModelManager](#). Yields a fatal error if we already have one module of that type (name). Typically, you want to set this flag to true in models where the Brain is your top-level (root) component; in models that contains objects for whom the Brain is a subcomponent, typically they will also have our modules as subcomponents, so we should not set this flag to true

Definition at line 168 of file Brain.C.

References `ModelComponent::addSubComponent()`, `SharedPtr< GistEstimator >::dynCastFrom()`, `SharedPtr< InferoTemporal >::dynCastFrom()`, `SharedPtr< WinnerTakeAll >::dynCastFrom()`, `SharedPtr< AttentionGuidanceMap >::dynCastFrom()`, `SharedPtr< TaskRelevanceMap >::dynCastFrom()`, `SharedPtr< SaliencyMap >::dynCastFrom()`, `SharedPtr< ShapeEstimator >::dynCastFrom()`, `SharedPtr< SimulationViewer >::dynCastFrom()`, `SharedPtr< SaccadeController >::dynCastFrom()`, `SharedPtr< VisualCortex >::dynCastFrom()`, `SharedPtr< Retina >::dynCastFrom()`, `SharedPtr< GistEstimator >::get()`, `SharedPtr< InferoTemporal >::get()`, `SharedPtr< WinnerTakeAll >::get()`, `SharedPtr< AttentionGuidanceMap >::get()`, `SharedPtr< TaskRelevanceMap >::get()`, `SharedPtr< SaliencyMap >::get()`, `SharedPtr< ShapeEstimator >::get()`, `SharedPtr< SimulationViewer >::get()`, `SharedPtr< SaccadeController >::get()`, `SharedPtr< VisualCortex >::get()`, `SharedPtr< Retina >::get()`, `itsGE`, `itsIT`, `itsRET`, `itsSC`, `itsSE`, `itsSM`, `itsSV`, `itsTRM`, `itsVC`, and `itsWTA`.

Referenced by `GetSaliency::GetSaliency()`, `StdBrain::start1()`, and `StdBrain::StdBrain()`.

### 10.37.3.2 void Brain::blinkSuppression (const bool on)

Turn blink suppression on/off.

For now, we only reset the WTA. If we have a [SaccadeController](#) module, we will call this internally during our [evolve\(\)](#) as appropriate; this method is made public just so that blink suppression may be forced externally too.

Definition at line 925 of file Brain.C.

References itsWTA.

Referenced by evolve().

### 10.37.3.3 void Brain::evolve (double & stime, [Point2D](#) & winner, int & ntarg, float & winV)

Evolve saliency map and winner-take-all.

#### Parameters:

**stime** current simulation time, in seconds

**winner** coordinates of new attended location; winner coords are -1 if there was no new winner (i.e., no shift of attention) If we have a [SaccadeController](#) module, winner will contain the current eye position rather than attention position.

**ntarg** number of new targets hit (or -1 if whole visual field visited)

**winV** saliency map voltage at winning location.

Definition at line 561 of file Brain.C.

References [blinkSuppression\(\)](#), [WTAwinner::boring](#), [WTAwinner::buildFromSMcoords\(\)](#), [SharedPtr< T >::dynCastFrom\(\)](#), [Image< byte >::freeMem\(\)](#), [SharedPtr< InferoTemporal >::get\(\)](#), [SharedPtr< ShapeEstimator >::get\(\)](#), [SharedPtr< SimulationViewer >::get\(\)](#), [SharedPtr< AttentionGuidanceMap >::get\(\)](#), [SharedPtr< TaskRelevanceMap >::get\(\)](#), [SharedPtr< VisualCortex >::get\(\)](#), [SharedPtr< SaccadeController >::get\(\)](#), [Image< T >::getHeight\(\)](#), [ModelParam< IORtype >::getVal\(\)](#), [ModelParam< int >::getVal\(\)](#), [ModelParam< bool >::getVal\(\)](#), [ModelParam< LevelSpec >::getVal\(\)](#), [ModelParam< double >::getVal\(\)](#), [ModelParam< float >::getVal\(\)](#), [Image< T >::getWidth\(\)](#), [Point2D::i](#), [Image< byte >::initialized\(\)](#), [IORdisc](#), [IORnone](#), [IORshapeEst](#), [IORshapeEstCM](#), [IORshapeEstSM](#), [itsBoringDelay](#), [itsBoringSMmv](#), [itsFOAradius](#), [itsIORtype](#), [itsIT](#), [itsLevelSpec](#), [itsMaxWinMv](#), [itsSaveWinnerFeatures](#), [itsSC](#), [itsSE](#), [itsSM](#), [itsSV](#), [itsTimeStep](#), [itsTooManyShifts](#), [itsTooMuchTime](#), [itsTRM](#), [itsUseRandom](#), [itsVC](#), [itsWTA](#), [Point2D::j](#), [LevelSpec::mapLevel\(\)](#), [WTAwinner::p](#), [saccadicSuppression\(\)](#), [WTAwinner::sv](#), [WTAwinner::t](#), [ModelManager::textLog\(\)](#), [uint](#), and [uint64](#).

Referenced by [GetSaliency::compute\(\)](#).



### 10.37.3.4 void Brain::exportOptions (const int *whatToExport*, const bool *recurse* = true) [virtual]

Export some of our [ModelParam](#) parameters as command-line-options.

See the base function in [ModelComponent.H](#) for info.

Reimplemented from [ModelComponent](#).

Definition at line 103 of file Brain.C.

References [ModelComponent::exportOptions\(\)](#), [itsBlankBlink](#), [itsFOARadius](#), [itsFoveaRadius](#), [itsIORtype](#), [itsLevelSpec](#), [itsSaveObjMask](#), [itsSaveWinnerFeatures](#), [itsTimeStep](#), [itsTooManyShifts](#), [itsTooMuchTime](#), [itsUseRandom](#), [OPTEXP\\_CORE](#), [OPTEXP\\_SAVE](#), and [ModelManager::requestOption\(\)](#).

### 10.37.3.5 void Brain::findFixation ([Point2D](#) & *fixation*, float & *value*) const

Finds the current fixation in AGM.

#### Parameters:

***fixation*** enter a reference to current fixation ie a dummy that returns the current fixation point

***relevance,enter*** a reference to value at current fixation ie a dummy that returns the value at current fixation

Definition at line 911 of file Brain.C.

### 10.37.3.6 [SharedPtr](#)< [AttentionGuidanceMap](#) > Brain::getAGM () const

Get the [AttentionGuidanceMap](#).

Definition at line 390 of file Brain.C.

References [SharedPtr](#)< [AttentionGuidanceMap](#) >::get().

Referenced by [SimulationViewerEyeMvt2::attentionShift\(\)](#), [SimulationViewerEyeMvt::eyeMovement\(\)](#), [SimulationViewerEyeMvt2::getTraj\(\)](#), [SimulationViewerEyeMvt::getTraj\(\)](#), and [SimulationViewerCompress::getTraj\(\)](#).

### 10.37.3.7 [SharedPtr](#)< [GistEstimator](#) > Brain::getGE () const

Get the [GistEstimator](#).

Definition at line 412 of file Brain.C.

References [SharedPtr](#)< [GistEstimator](#) >::get(), and [itsGE](#).

Referenced by [StdBrain::start1\(\)](#).

### 10.37.3.8 `const Dims & Brain::getInputDims () const`

Get input dimensions.

Definition at line 899 of file Brain.C.

References `Image< PixRGB< byte > >::getDims()`.

Referenced by `TaskRelevanceMapStd::update()`.

### 10.37.3.9 `SharedPtr< IneroTemporal > Brain::getIT () const`

Get the [IneroTemporal](#).

Definition at line 405 of file Brain.C.

References `SharedPtr< IneroTemporal >::get()`, and its `IT`.

### 10.37.3.10 `Image< byte > Brain::getObjectMask () const`

Returns binary mask of currently attended object.

Definition at line 895 of file Brain.C.

### 10.37.3.11 `Image< PixRGB< byte > > Brain::getRawInput () const`

Get whatever raw input we have received before passing it through [Retina](#).

Definition at line 903 of file Brain.C.

Referenced by `SimulationViewerCompress::getTraj()`.

### 10.37.3.12 `SharedPtr< Retina > Brain::getRET () const`

Get the [Retina](#).

Definition at line 341 of file Brain.C.

References `SharedPtr< Retina >::get()`, and its `RET`.

Referenced by `VisualBuffer::bufferToRetinal()`, `EyeTrackerSaccadeController::evolve()`, `SimulationViewerEyeMvt2::eyeMovement()`, `SimulationViewerCompress::getTraj()`, `SimulationViewerEyeMvt2::newInputFrame()`, `VisualBuffer::retinalToBuffer()`, `StdBrain::start1()`, and `SimulationViewerEyeMvt2::start1()`.

### 10.37.3.13 `SharedPtr< SaccadeController > Brain::getSC () const`

Get the [SaccadeController](#).

Definition at line 355 of file Brain.C.

References `SharedPtr< SaccadeController >::get()`, and `itsSC`.

Referenced by `RetinaStd::doGetOutput()`, `TaskRelevanceMapKillStatic::start1()`, `StdBrain::start1()`, `SimulationViewerEyeMvt2::start1()`, and `SimulationViewerEyeMvt::start1()`.

#### 10.37.3.14 `SharedPtr< ShapeEstimator > Brain::getSE () const`

Get the [ShapeEstimator](#).

Definition at line 369 of file Brain.C.

References `SharedPtr< ShapeEstimator >::get()`, and `itsSE`.

Referenced by `StdBrain::start1()`, and `TaskRelevanceMapStd::update()`.

#### 10.37.3.15 `SharedPtr< SaliencyMap > Brain::getSM () const`

Get the [SaliencyMap](#).

Definition at line 376 of file Brain.C.

References `SharedPtr< SaliencyMap >::get()`, and `itsSM`.

Referenced by `SimulationViewerEyeMvt2::attentionShift()`, `SimulationViewerStd::drawMegaCombo()`, `SimulationViewerStd::getTraj()`, `SimulationViewerCompress::getTraj()`, and `SimulationViewerEyeMvt2::newInputFrame()`.

#### 10.37.3.16 `SharedPtr< SimulationViewer > Brain::getSV () const`

Get the [SimulationViewer](#).

Definition at line 362 of file Brain.C.

References `SharedPtr< SimulationViewer >::get()`, and `itsSV`.

Referenced by `StdBrain::start1()`.

#### 10.37.3.17 `SharedPtr< TaskRelevanceMap > Brain::getTRM () const`

Get the [TaskRelevanceMap](#).

Definition at line 383 of file Brain.C.

References `SharedPtr< TaskRelevanceMap >::get()`, and `itsTRM`.

Referenced by `SimulationViewerEyeMvt2::attentionShift()`, `SimulationViewerStd::drawMegaCombo()`, `SimulationViewerStd::getTraj()`, and `SimulationViewer-`

Compress::getTraj(), and StdBrain::start1().

#### 10.37.3.18 **SharedPtr< VisualCortex > Brain::getVC () const**

Get the [VisualCortex](#).

Definition at line 348 of file Brain.C.

References [SharedPtr< VisualCortex >::get\(\)](#), and [itsVC](#).

Referenced by [GetSaliency::compute\(\)](#), [SimulationViewerStd::drawMegaCombo\(\)](#), [SimulationViewerStd::drawMotionSixPack\(\)](#), [SimulationViewerEyeMvt::eye-Movement\(\)](#), [SimulationViewerCompress::getTraj\(\)](#), [TaskRelevanceMapKill-Static::start1\(\)](#), and [StdBrain::start1\(\)](#).

#### 10.37.3.19 **Image< byte > Brain::getVisualField () const**

Returns current visual field.

The visual field starts empty and fills in with each FOA shift

Definition at line 907 of file Brain.C.

#### 10.37.3.20 **SharedPtr< WinnerTakeAll > Brain::getWTA () const**

Get the [WinnerTakeAll](#).

Definition at line 397 of file Brain.C.

References [SharedPtr< WinnerTakeAll >::get\(\)](#), and [itsWTA](#).

#### 10.37.3.21 **bool Brain::hasModule (const char \* modtype) const**

Returns true if the corresponding module has been added with [addModule\(\)](#).

Definition at line 419 of file Brain.C.

References [SharedPtr< GistEstimator >::get\(\)](#), [SharedPtr< IneroTemporal >::get\(\)](#), [SharedPtr< WinnerTakeAll >::get\(\)](#), [SharedPtr< AttentionGuidanceMap >::get\(\)](#), [SharedPtr< TaskRelevanceMap >::get\(\)](#), [SharedPtr< SaliencyMap >::get\(\)](#), [SharedPtr< ShapeEstimator >::get\(\)](#), [SharedPtr< SimulationViewer >::get\(\)](#), [SharedPtr< SaccadeController >::get\(\)](#), [SharedPtr< VisualCortex >::get\(\)](#), [SharedPtr< Retina >::get\(\)](#), [itsGE](#), [itsIT](#), [itsRET](#), [itsSC](#), [itsSE](#), [itsSM](#), [itsSV](#), [itsTRM](#), [itsVC](#), and [itsWTA](#).

Referenced by [RetinaStd::doGetOutput\(\)](#), and [StdBrain::start1\(\)](#).

**10.37.3.22** `void Brain::input (const Image< PixRGB< byte > > & inp, const Image< byte > & targets, const Image< byte > & clipmask = Image< byte >())`

Load a new frame and pass it through visual cortex.

This will yield a new input to the saliency map.

**Parameters:**

*inp* input color image

*targets* binary map of the same size as the input, in which targets for the focus of attention are 255 and the rest is zero. Pass an uninitialized image if you have no targets to look for.

*clipMask* binary map of the same size as the input, in which areas that are to be considered for processing for attention are 255, and areas that are to be "clipped" from the Saliency Map are 0. An uninitialized image means: don't use a mask.

Definition at line 466 of file Brain.C.

References `obtainOutput()`, and `processInput()`.

Referenced by `GetSaliency::compute()`.

**10.37.3.23** `SharedPtr< ModelComponent > Brain::module (const char * modtype) const`

Get a module; see shorthands below.

Yields a `SharedPtr` to NULL if the corresponding module has not been added using `addModule()`, and throws a fatal error if `modtype` is unknown

Definition at line 322 of file Brain.C.

References `itsGE`, `itsIT`, `itsRET`, `itsSC`, `itsSE`, `itsSM`, `itsSV`, `itsTRM`, `itsVC`, and `itsWTA`.

**10.37.3.24** `void Brain::obtainOutput ()`

Get the output from `VisualCortex` & Co. (called from input).

Definition at line 533 of file Brain.C.

References `Image< T >::clear()`, `SharedPtr< SimulationViewer >::get()`, `SharedPtr< WinnerTakeAll >::get()`, `SharedPtr< TaskRelevanceMap >::get()`, `SharedPtr< SaliencyMap >::get()`, `Image< T >::getDims()`, `Image< PixRGB< byte > >::getDims()`, `Image< T >::isSameSize()`, `Image< byte >::isSameSize()`, `itsSM`, `itsSV`, `itsTRM`, `itsVC`, `itsWTA`, `NO_INIT`, and `ZEROS`.

Referenced by `input()`.

**10.37.3.25** `void Brain::processInput (const Image< PixRGB< byte > > & inp, const Image< byte > & targets, const Image< byte > & clipmask = Image< byte >())`

Deliver output to VC & Co (called from `input`, see there for details).

Definition at line 474 of file `Brain.C`.

References `Image< byte >::countParticles()`, `Image< byte >::freeMem()`, `SharedPtr< GistEstimator >::get()`, `SharedPtr< SaccadeController >::get()`, `SharedPtr< VisualCortex >::get()`, `SharedPtr< Retina >::get()`, `SharedPtr< SimulationViewer >::get()`, `Image< PixRGB< byte > >::getDims()`, `ModelParam< bool >::getVal()`, `Image< T >::initialized()`, `Image< PixRGB< byte > >::initialized()`, `itsBlankBlink`, `itsGE`, `itsRET`, `itsSC`, `itsSV`, `itsVC`, `Image< byte >::normalize()`, `ModelManager::textLog()`, and `ZEROS`.

Referenced by `input()`.

**10.37.3.26** `void Brain::removeModule (const char * modtype)`

Remove a previously added module.

Yields a fatal error if no such module was previously added

Definition at line 254 of file `Brain.C`.

References `SharedPtr< GistEstimator >::get()`, `SharedPtr< IneroTemporal >::get()`, `SharedPtr< WinnerTakeAll >::get()`, `SharedPtr< AttentionGuidanceMap >::get()`, `SharedPtr< TaskRelevanceMap >::get()`, `SharedPtr< SaliencyMap >::get()`, `SharedPtr< ShapeEstimator >::get()`, `SharedPtr< SimulationViewer >::get()`, `SharedPtr< SaccadeController >::get()`, `SharedPtr< VisualCortex >::get()`, `SharedPtr< Retina >::get()`, `ModelComponent::hasSubComponent()`, `itsGE`, `itsIT`, `itsRET`, `itsSC`, `itsSE`, `itsSM`, `itsSV`, `itsTRM`, `itsVC`, `itsWTA`, `ModelComponent::removeSubComponent()`, `SharedPtr< GistEstimator >::reset()`, `SharedPtr< IneroTemporal >::reset()`, `SharedPtr< WinnerTakeAll >::reset()`, `SharedPtr< AttentionGuidanceMap >::reset()`, `SharedPtr< TaskRelevanceMap >::reset()`, `SharedPtr< SaliencyMap >::reset()`, `SharedPtr< ShapeEstimator >::reset()`, `SharedPtr< SimulationViewer >::reset()`, `SharedPtr< SaccadeController >::reset()`, `SharedPtr< VisualCortex >::reset()`, and `SharedPtr< Retina >::reset()`.

**10.37.3.27** `void Brain::reset (const bool recurse = true) [virtual]`

Reset Brain and propagate the reset signal to the sub-components.

See the base function in [ModelComponent.H](#) for info.

Reimplemented from [ModelComponent](#).

Definition at line 451 of file Brain.C.

References `Image< byte >::freeMem()`, `Image< PixRGB< byte > >::freeMem()`, and `ModelComponent::reset()`.

### 10.37.3.28 double Brain::resetTime ()

Reset the internal time.

Definition at line 891 of file Brain.C.

### 10.37.3.29 void Brain::saccadicSuppression (const bool on)

Turn saccadic suppression on/off.

For now, this is a pass-through to `SM::saccadicSuppression()`, so see there for usage. In addition, we reset our static-detection buffer and also the WTA. If we have a [Saccade-Controller](#) module, we will call this internally during our `evolve()`, as appropriate; this method is made public just so that saccadic suppression may be forced externally too.

Definition at line 915 of file Brain.C.

References `itsSM`, and `itsWTA`.

Referenced by `evolve()`.

### 10.37.3.30 void Brain::saveResults (const SharedPtr< OutputFrameSeries > & ofs)

Save our internals and outputs from our modules.

Definition at line 984 of file Brain.C.

References `SharedPtr< WinnerTakeAll >::get()`, `SharedPtr< AttentionGuidanceMap >::get()`, `SharedPtr< TaskRelevanceMap >::get()`, `SharedPtr< SaliencyMap >::get()`, `SharedPtr< SimulationViewer >::get()`, `SharedPtr< VisualCortex >::get()`, `ModelParam< bool >::getVal()`, `Image< byte >::initialized()`, `itsSaveObjMask`, `itsSM`, `itsSV`, `itsTRM`, `itsVC`, and `itsWTA`.

### 10.37.3.31 void Brain::start1 () [protected, virtual]

do some additional config at start time; see [ModelComponent.H](#)

Reimplemented from [ModelComponent](#).

Reimplemented in [StdBrain](#).

Definition at line 127 of file Brain.C.

References `SharedPtr< ShapeEstimator >::get()`, `ModelParam< IORtype >::getVal()`, `IORauto`, `IORdisc`, `IORshapeEst`, `IORshapeEstCM`, `IORshapeEstSM`, `itsIORtype`, `itsSE`, `SEMconspicuityMap`, `SEMfeatureMap`, `SEMsaliencyMap`, `ModelParam< IORtype >::setVal()`, and `ShapeEstimatorMode`.

Referenced by `StdBrain::start1()`.

## 10.37.4 Member Data Documentation

### 10.37.4.1 `SharedPtr<AttentionGuidanceMap> Brain::itsAGM` [protected]

`AttentionGuidanceMap` Module.

Definition at line 332 of file Brain.H.

### 10.37.4.2 `ModelParam<bool> Brain::itsBlankBlink` [protected]

Blank out visual inputs during blinks?

Definition at line 302 of file Brain.H.

Referenced by `exportOptions()`, and `processInput()`.

### 10.37.4.3 `ModelParam<double> Brain::itsBoringDelay` [protected]

Inter-covert-shift delay (in ms) that makes a shift of attention boring.

Definition at line 291 of file Brain.H.

Referenced by `evolve()`.

### 10.37.4.4 `ModelParam<float> Brain::itsBoringSMmv` [protected]

Minimum saliency map depolarization (in mV) to make a covert attention shift not boring

Definition at line 295 of file Brain.H.

Referenced by `evolve()`.

### 10.37.4.5 `ModelParam<int> Brain::itsFOAradius` [protected]

radius of the FOA, in pixels



Definition at line 278 of file Brain.H.

Referenced by `evolve()`, and `exportOptions()`.

#### 10.37.4.6 [ModelParam<int>](#) [Brain::itsFoveaRadius](#) [protected]

radius of the Fovea, in pixels

Definition at line 281 of file Brain.H.

Referenced by `exportOptions()`.

#### 10.37.4.7 [SharedPtr<GistEstimator>](#) [Brain::itsGE](#) [protected]

[GistEstimator](#) Module.

Definition at line 335 of file Brain.H.

Referenced by `addModule()`, `getGE()`, `hasModule()`, `module()`, `processInput()`, and `removeModule()`.

#### 10.37.4.8 [ModelParam<IORtype>](#) [Brain::itsIORtype](#) [protected]

type of IOR to use. See [IORtypes.H](#)

Definition at line 272 of file Brain.H.

Referenced by `evolve()`, `exportOptions()`, and `start1()`.

#### 10.37.4.9 [SharedPtr<InferoTemporal>](#) [Brain::itsIT](#) [protected]

[InferoTemporal](#) Module.

Definition at line 334 of file Brain.H.

Referenced by `addModule()`, `evolve()`, `getIT()`, `hasModule()`, `module()`, and `removeModule()`.

#### 10.37.4.10 [ModelParam<LevelSpec>](#) [Brain::itsLevelSpec](#) [protected]

Specification of our image processing center-surround levels.

The only thing that we care about here is the saliency map level

Definition at line 288 of file Brain.H.

Referenced by `evolve()`, and `exportOptions()`.

**10.37.4.11** [ModelParam](#)<float> [Brain::itsMaxWinMv](#) [protected]

If the saliency map voltage at winning location gets higher than this (in mV), trigger some global inhibition

Definition at line 299 of file Brain.H.

Referenced by `evolve()`.

**10.37.4.12** [SharedPtr](#)<[Retina](#)> [Brain::itsRET](#) [protected]

[Retina](#) module.

Definition at line 325 of file Brain.H.

Referenced by `addModule()`, `getRET()`, `hasModule()`, `module()`, `processInput()`, and `removeModule()`.

**10.37.4.13** [ModelParam](#)<bool> [Brain::itsSaveObjMask](#) [protected]

Save our internals when `saveResults()` is called?

Definition at line 305 of file Brain.H.

Referenced by `exportOptions()`, and `saveResults()`.

**10.37.4.14** [ModelParam](#)<bool> [Brain::itsSaveWinnerFeatures](#)  
[protected]

Save vector of features at each attended location?

Definition at line 314 of file Brain.H.

Referenced by `evolve()`, and `exportOptions()`.

**10.37.4.15** [SharedPtr](#)<[SaccadeController](#)> [Brain::itsSC](#) [protected]

[SaccadeController](#) module.

Definition at line 327 of file Brain.H.

Referenced by `addModule()`, `evolve()`, `getSC()`, `hasModule()`, `module()`, `processInput()`, and `removeModule()`.

**10.37.4.16** [SharedPtr](#)<[ShapeEstimator](#)> [Brain::itsSE](#) [protected]

[ShapeEstimator](#) Module.

Definition at line 329 of file Brain.H.

Referenced by `addModule()`, `evolve()`, `getSE()`, `hasModule()`, `module()`, `removeModule()`, and `start1()`.

#### 10.37.4.17 `SharedPtr<SaliencyMap> Brain::itsSM` [protected]

`SaliencyMap` Module.

Definition at line 330 of file Brain.H.

Referenced by `addModule()`, `evolve()`, `getSM()`, `hasModule()`, `module()`, `obtainOutput()`, `removeModule()`, `saccadicSuppression()`, and `saveResults()`.

#### 10.37.4.18 `SharedPtr<SimulationViewer> Brain::itsSV` [protected]

`SimulationViewer` module.

Definition at line 328 of file Brain.H.

Referenced by `addModule()`, `evolve()`, `getSV()`, `hasModule()`, `module()`, `obtainOutput()`, `processInput()`, `removeModule()`, and `saveResults()`.

#### 10.37.4.19 `ModelParam<double> Brain::itsTimeStep` [protected]

Simulation time step, in seconds.

Definition at line 284 of file Brain.H.

Referenced by `evolve()`, and `exportOptions()`.

#### 10.37.4.20 `ModelParam<int> Brain::itsTooManyShifts` [protected]

Exit after a given number of attention shifts (if not zero).

Definition at line 308 of file Brain.H.

Referenced by `evolve()`, and `exportOptions()`.

#### 10.37.4.21 `ModelParam<float> Brain::itsTooMuchTime` [protected]

Exit after a given time in seconds (if not zero).

Definition at line 311 of file Brain.H.

Referenced by `evolve()`, and `exportOptions()`.

**10.37.4.22**   [SharedPtr<TaskRelevanceMap>](#)   [Brain::itsTRM](#)   [protected]

[TaskRelevanceMap](#) Module.

Definition at line 331 of file Brain.H.

Referenced by [addModule\(\)](#), [evolve\(\)](#), [getTRM\(\)](#), [hasModule\(\)](#), [module\(\)](#), [obtainOutput\(\)](#), [removeModule\(\)](#), and [saveResults\(\)](#).

**10.37.4.23**   [ModelParam<bool>](#)   [Brain::itsUseRandom](#)   [protected]

add jitter to winning location?

Definition at line 275 of file Brain.H.

Referenced by [evolve\(\)](#), and [exportOptions\(\)](#).

**10.37.4.24**   [SharedPtr<VisualCortex>](#)   [Brain::itsVC](#)   [protected]

[VisualCortex](#) module.

Definition at line 326 of file Brain.H.

Referenced by [addModule\(\)](#), [evolve\(\)](#), [getVC\(\)](#), [hasModule\(\)](#), [module\(\)](#), [obtainOutput\(\)](#), [processInput\(\)](#), [removeModule\(\)](#), and [saveResults\(\)](#).

**10.37.4.25**   [SharedPtr<WinnerTakeAll>](#)   [Brain::itsWTA](#)   [protected]

[WinnerTakeAll](#) Module.

Definition at line 333 of file Brain.H.

Referenced by [addModule\(\)](#), [blinkSuppression\(\)](#), [evolve\(\)](#), [getWTA\(\)](#), [hasModule\(\)](#), [module\(\)](#), [obtainOutput\(\)](#), [removeModule\(\)](#), [saccadicSuppression\(\)](#), and [saveResults\(\)](#).

The documentation for this class was generated from the following files:

- [Brain.H](#)
- [Brain.C](#)

## 10.38 bug Class Reference

This gets thrown by the FATAL logs.

```
#include <log.H>
```

### 10.38.1 Detailed Description

This gets thrown by the FATAL logs.

Definition at line 127 of file log.H.

The documentation for this class was generated from the following file:

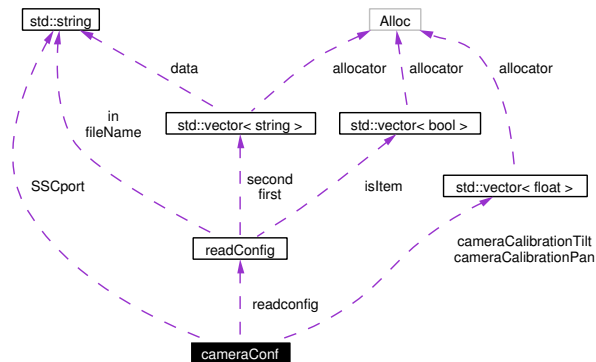
- [log.H](#)

## 10.39 cameraConf Class Reference

This class uses [readConfig](#) to extract config parameters for [CameraControl](#).

```
#include <cameraConf.H>
```

Collaboration diagram for cameraConf:



### Public Member Functions

- [cameraConf](#) ([string](#) fileName)  
*default constructor, open config fileName and read*
- [cameraConf](#) ()  
*default constructor, open config "camera.conf" and read*
- [~cameraConf](#) ()  
*default destructor*

### Public Attributes

- unsigned int [Xpixel](#)  
*size of field of view in pixels*
- unsigned int [Ypixel](#)  
*size of field of view in pixels*
- float [Xfield](#)  
*size of field of view in degrees*

- float [Yfield](#)  
*size of field of view in degrees*
- float [pixelSizeMM](#)  
*pixel size in millimeters*
- float [focalLengthMM](#)  
*focal length in millimeters*
- float [panOffsetMM](#)  
*pan offset in millimeters*
- float [tiltOffsetMM](#)  
*tilt offset in millimeters*
- float [Xcenter](#)  
*camera center in degrees*
- float [Ycenter](#)  
*camera center in degrees*
- float [servoPixels](#)  
*servo pixels per degree*
- float [travelSpeed](#)  
*default travel speed from 0 to 1*
- int [travelTime](#)  
*the time it takes to complete a 90 degree travel in milliseconds*
- int [minTravelTime](#)  
*minimum travel time*
- unsigned int [startCenterCamera](#)  
*center camera at start 1 = yes*
- unsigned int [finishCenterCamera](#)  
*center camera at finish*
- unsigned int [SSCXconnector](#)  
*servo connections on SSC*

- unsigned int [SSCYconnector](#)  
*servo connections on SSC*
- unsigned int [SSCbaud](#)  
*baud tranfer rate 2400 or 9600 only*
- string [SSCport](#)  
*SSC serial port device;.*
- float [XlimitStart](#)  
*X Travel Limit in degrees.*
- float [XlimitEnd](#)  
*X Travel Limit in degrees.*
- float [YlimitStart](#)  
*Y Travel Limit in degrees.*
- float [YlimitEnd](#)  
*Y Travel Limit in degrees.*
- float [fieldAdjustmentX](#)  
*manual field adjustment*
- float [fieldAdjustmentY](#)  
*manual field adjustment*
- float [pixelField](#)  
*field size in pixels*
- float [panCalibrate](#)  
*Pan calibration in degrees.*
- float [tiltCalibrate](#)  
*Tilt calibration in degrees.*
- float [minMove](#)  
*minimum distance to move in seccade*
- [std::vector](#)< float > **cameraCalibrationPan**
- [std::vector](#)< float > **cameraCalibrationTilt**



### 10.39.1 Detailed Description

This class uses [readConfig](#) to extract config parameters for [CameraControl](#).

This class reads config parameters from "camera.conf" or another specified config file and organizes them for use in [CameraControl](#).

Definition at line 12 of file cameraConf.H.

### 10.39.2 Constructor & Destructor Documentation

#### 10.39.2.1 cameraConf::cameraConf ([string](#) *fileName*)

default constructor, open config *fileName* and read

Definition at line 12 of file cameraConf.C.

References [readConfig::openFile\(\)](#).

#### 10.39.2.2 cameraConf::cameraConf ()

default constructor, open config "camera.conf" and read

Definition at line 7 of file cameraConf.C.

References [readConfig::openFile\(\)](#).

#### 10.39.2.3 cameraConf::~~cameraConf ()

default destructor

Definition at line 97 of file cameraConf.C.

### 10.39.3 Member Data Documentation

#### 10.39.3.1 float [cameraConf::fieldAdjustmentX](#)

manual field adjustment

Definition at line 74 of file cameraConf.H.

Referenced by [CameraControl::moveCamXYFrame\(\)](#).

#### 10.39.3.2 float [cameraConf::fieldAdjustmentY](#)

manual field adjustment

Definition at line 76 of file cameraConf.H.

Referenced by CameraControl::moveCamXYFrame().

#### **10.39.3.3   unsigned int [cameraConf::finishCenterCamera](#)**

center camera at finish

Definition at line 56 of file cameraConf.H.

#### **10.39.3.4   float [cameraConf::focalLengthMM](#)**

focal length in millimeters

Definition at line 35 of file cameraConf.H.

#### **10.39.3.5   float [cameraConf::minMove](#)**

minimum distance to move in seccade

Definition at line 84 of file cameraConf.H.

Referenced by CameraControl::moveCamXYFrame().

#### **10.39.3.6   int [cameraConf::minTravelTime](#)**

minimum travel time

Definition at line 52 of file cameraConf.H.

Referenced by CameraControl::moveCamTPFrame(), and CameraControl::moveCamXYFrame().

#### **10.39.3.7   float [cameraConf::panCalibrate](#)**

Pan calibration in degrees.

Definition at line 80 of file cameraConf.H.

#### **10.39.3.8   float [cameraConf::panOffsetMM](#)**

pan offset in millimeters

Definition at line 37 of file cameraConf.H.

**10.39.3.9 float [cameraConf::pixelField](#)**

field size in pixels

Definition at line 78 of file cameraConf.H.

**10.39.3.10 float [cameraConf::pixelSizeMM](#)**

pixel size in millimeters

Definition at line 33 of file cameraConf.H.

**10.39.3.11 float [cameraConf::servoPixels](#)**

servo pixels per degree

Definition at line 46 of file cameraConf.H.

Referenced by [CameraControl::centerCamera\(\)](#), [CameraControl::moveCamTPFrame\(\)](#), and [CameraControl::moveCamXYFrame\(\)](#).

**10.39.3.12 unsigned int [cameraConf::SSCbaud](#)**

baud tranfer rate 2400 or 9600 only

Definition at line 62 of file cameraConf.H.

**10.39.3.13 string [cameraConf::SSCport](#)**

[SSC](#) serial port device;.

Definition at line 64 of file cameraConf.H.

**10.39.3.14 unsigned int [cameraConf::SSCXconnector](#)**

servo connections on [SSC](#)

Definition at line 58 of file cameraConf.H.

Referenced by [CameraControl::panAbsolute\(\)](#), and [CameraControl::panRelative\(\)](#).

**10.39.3.15 unsigned int [cameraConf::SSCYconnector](#)**

servo connections on [SSC](#)

Definition at line 60 of file cameraConf.H.

Referenced by `CameraControl::tiltAbsolute()`, and `CameraControl::tiltRelative()`.

#### **10.39.3.16 unsigned int `cameraConf::startCenterCamera`**

center camera at start 1 = yes

Definition at line 54 of file `cameraConf.H`.

#### **10.39.3.17 float `cameraConf::tiltCalibrate`**

Tilt calibration in degrees.

Definition at line 82 of file `cameraConf.H`.

#### **10.39.3.18 float `cameraConf::tiltOffsetMM`**

tilt offset in millimeters

Definition at line 39 of file `cameraConf.H`.

#### **10.39.3.19 float `cameraConf::travelSpeed`**

default travel speed from 0 to 1

Definition at line 48 of file `cameraConf.H`.

#### **10.39.3.20 int `cameraConf::travelTime`**

the time it takes to complete a 90 degree travel in milliseconds

Definition at line 50 of file `cameraConf.H`.

Referenced by `CameraControl::moveCamTPFrame()`, and `CameraControl::moveCam-XYFrame()`.

#### **10.39.3.21 float `cameraConf::Xcenter`**

camera center in degrees

Definition at line 42 of file `cameraConf.H`.

Referenced by `CameraControl::CameraControl()`, and `CameraControl::center-Camera()`.

**10.39.3.22 float cameraConf::Xfield**

size of field of view in degrees

Definition at line 29 of file cameraConf.H.

Referenced by CameraControl::moveCamXYFrame().

**10.39.3.23 float cameraConf::XlimitEnd**

X Travel Limit in degrees.

Definition at line 68 of file cameraConf.H.

Referenced by CameraControl::moveCamTPFrame(), CameraControl::moveCamXYFrame(), CameraControl::panAbsolute(), and CameraControl::panRelative().

**10.39.3.24 float cameraConf::XlimitStart**

X Travel Limit in degrees.

Definition at line 66 of file cameraConf.H.

Referenced by CameraControl::moveCamTPFrame(), CameraControl::moveCamXYFrame(), CameraControl::panAbsolute(), and CameraControl::panRelative().

**10.39.3.25 unsigned int cameraConf::Xpixel**

size of field of view in pixels

Definition at line 25 of file cameraConf.H.

Referenced by CameraControl::moveCamXYFrame(), and CameraControl::setImageSize().

**10.39.3.26 float cameraConf::Ycenter**

camera center in degrees

Definition at line 44 of file cameraConf.H.

Referenced by CameraControl::CameraControl(), and CameraControl::centerCamera().

**10.39.3.27 float cameraConf::Yfield**

size of field of view in degrees

Definition at line 31 of file cameraConf.H.

Referenced by CameraControl::moveCamXYFrame().

#### **10.39.3.28 float cameraConf::YlimitEnd**

Y Travel Limit in degrees.

Definition at line 72 of file cameraConf.H.

Referenced by CameraControl::moveCamTPFrame(), CameraControl::moveCamXYFrame(), CameraControl::tiltAbsolute(), and CameraControl::tiltRelative().

#### **10.39.3.29 float cameraConf::YlimitStart**

Y Travel Limit in degrees.

Definition at line 70 of file cameraConf.H.

Referenced by CameraControl::moveCamTPFrame(), CameraControl::moveCamXYFrame(), CameraControl::tiltAbsolute(), and CameraControl::tiltRelative().

#### **10.39.3.30 unsigned int cameraConf::Ypixel**

size of field of view in pixels

Definition at line 27 of file cameraConf.H.

Referenced by CameraControl::moveCamXYFrame(), and CameraControl::setImageSize().

The documentation for this class was generated from the following files:

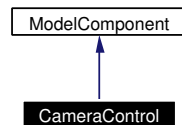
- [cameraConf.H](#)
- [cameraConf.C](#)

## 10.40 CameraControl Class Reference

This class is an interface to pan, tilt, and otherwise move the camera.

```
#include <CameraControl.H>
```

Inheritance diagram for CameraControl:



Collaboration diagram for CameraControl:

### Public Member Functions

- [CameraControl](#) ([ModelManager](#) &mgr, const [std::string](#) &descrName="Camera Controller", const [std::string](#) &tagName="CameraControl", int setPort=0, bool setCam=true, int pPort=0, int tPort=1, int Port=1)  
*Initialize a camera control object.*
- [~CameraControl](#) ()  
*Destructor.*
- float [moveCamXYFrame](#) (int X, int Y, int camNumber=0, bool simluate=false)  
*move the camera to a point X,Y in an image*
- float [moveCamTPFrame](#) (float theta, float phi, int camNumber=0)  
*center the camera*
- void [centerCamera](#) (int camNumber=0)  
*center the camera*
- void [panRelative](#) (float doPan)  
*pan the camera a certain amount relative to the current pos in deg.*
- void [tiltRelative](#) (float doTilt)  
*tilt the camera a certain amount relative to the current pos in deg.*
- void [panAbsolute](#) (float doPan)  
*pan the camora to an absolute position in deg.*

- void [tiltAbsolute](#) (float doTilt)  
*tilt the camera to an absolute position in deg.*
- void [setImageSize](#) (int X, int Y)  
*set the size of the image for moveCamXYFrame if not 640x480*
- void [useCalibration](#) (bool useIt)  
*tells if calibration is to be used to different cameras*
- float [getCurrentPan](#) ()  
*get the current position pan of the camera in deg.*
- float [getCurrentTilt](#) ()  
*get the current position tilt of the camera in deg.*
- float [getSimPan](#) ()  
*get the current position pan of the camera in deg.*
- float [getSimTilt](#) ()  
*get the current position tilt of the camera in deg.*

### 10.40.1 Detailed Description

This class is an interface to pan, tilt, and otherwise move the camera.

Definition at line 10 of file CameraControl.H.

### 10.40.2 Constructor & Destructor Documentation

- 10.40.2.1** [CameraControl::CameraControl](#) ([ModelManager](#) & mgr, const [std::string](#) & descrName = "Camera Controller", const [std::string](#) & tagName = "CameraControl", int setPort = 0, bool setCam = true, int pPort = 0, int tPort = 1, int Port = 1)

Initialize a camera control object.

Definition at line 10 of file CameraControl.C.

References [ModelComponent::addSubComponent\(\)](#), [centerCamera\(\)](#), [cameraConf::Xcenter](#), and [cameraConf::Ycenter](#).



### 10.40.2.2 CameraControl::~~CameraControl ()

Destructor.

Definition at line 32 of file CameraControl.C.

## 10.40.3 Member Function Documentation

### 10.40.3.1 void CameraControl::centerCamera (int *camNumber* = 0)

center the camera

Definition at line 185 of file CameraControl.C.

References cameraConf::cameraCalibrationPan, cameraConf::cameraCalibrationTilt, cameraConf::servoPixels, cameraConf::Xcenter, and cameraConf::Ycenter.

Referenced by CameraControl().

### 10.40.3.2 float CameraControl::getCurrentPan ()

get the current position pan of the camera in deg.

Definition at line 259 of file CameraControl.C.

Referenced by panRelative().

### 10.40.3.3 float CameraControl::getCurrentTilt ()

get the current position tilt of the camera in deg.

Definition at line 267 of file CameraControl.C.

Referenced by tiltRelative().

### 10.40.3.4 float CameraControl::getSimPan ()

get the current position pan of the camera in deg.

Definition at line 275 of file CameraControl.C.

### 10.40.3.5 float CameraControl::getSimTilt ()

get the current position tilt of the camera in deg.

Definition at line 282 of file CameraControl.C.

#### 10.40.3.6 **float CameraControl::moveCamTPFrame (float *theta*, float *phi*, int *camNumber* = 0)**

center the camera

Definition at line 125 of file CameraControl.C.

References cameraConf::cameraCalibrationPan, cameraConf::cameraCalibrationTilt, cameraConf::minTravelTime, cameraConf::servoPixels, cameraConf::travelTime, cameraConf::XlimitEnd, cameraConf::XlimitStart, cameraConf::YlimitEnd, and cameraConf::YlimitStart.

#### 10.40.3.7 **float CameraControl::moveCamXYFrame (int *X*, int *Y*, int *camNumber* = 0, bool *simluate* = false)**

move the camera to a point X,Y in an image

Definition at line 37 of file CameraControl.C.

References abs(), ASSERT, cameraConf::cameraCalibrationPan, cameraConf::cameraCalibrationTilt, cameraConf::fieldAdjustmentX, cameraConf::fieldAdjustmentY, cameraConf::minMove, cameraConf::minTravelTime, cameraConf::servoPixels, cameraConf::travelTime, cameraConf::Xfield, cameraConf::XlimitEnd, cameraConf::XlimitStart, cameraConf::Xpixel, Y, cameraConf::Yfield, cameraConf::YlimitEnd, cameraConf::YlimitStart, and cameraConf::Ypixel.

#### 10.40.3.8 **void CameraControl::panAbsolute (float *doPan*)**

pan the camora to an absolute position in deg.

Definition at line 226 of file CameraControl.C.

References ASSERT, cameraConf::SSCXconnector, cameraConf::XlimitEnd, and cameraConf::XlimitStart.

#### 10.40.3.9 **void CameraControl::panRelative (float *doPan*)**

pan the camera a certain amount relative to the current pos in deg.

Definition at line 209 of file CameraControl.C.

References ASSERT, getCurrentPan(), cameraConf::SSCXconnector, cameraConf::XlimitEnd, and cameraConf::XlimitStart.

**10.40.3.10 void CameraControl::setImageSize (int X, int Y)**

set the size of the image for moveCamXYFrame if not 640x480

Definition at line 244 of file CameraControl.C.

References cameraConf::Xpixel, Y, and cameraConf::Ypixel.

**10.40.3.11 void CameraControl::tiltAbsolute (float *doTilt*)**

tilt the camera to an absolute position in deg.

Definition at line 235 of file CameraControl.C.

References ASSERT, cameraConf::SSCYconnector, cameraConf::YlimitEnd, and cameraConf::YlimitStart.

**10.40.3.12 void CameraControl::tiltRelative (float *doTilt*)**

tilt the camera a certain amount relative to the current pos in deg.

Definition at line 218 of file CameraControl.C.

References ASSERT, getCurrentTilt(), cameraConf::SSCYconnector, cameraConf::YlimitEnd, and cameraConf::YlimitStart.

**10.40.3.13 void CameraControl::useCalibration (bool *useIt*)**

tells if calibration is to be used to different cameras

Definition at line 252 of file CameraControl.C.

The documentation for this class was generated from the following files:

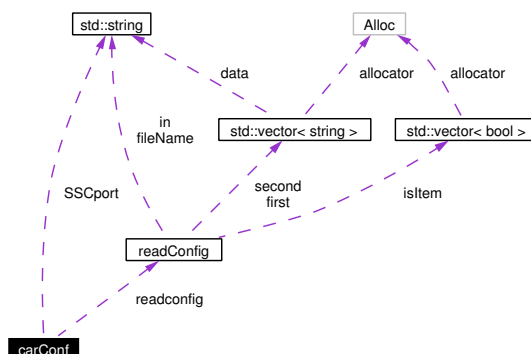
- [CameraControl.H](#)
- [CameraControl.C](#)

## 10.41 carConf Class Reference

This class uses [readConfig](#) to extract config parameters for [CarControl](#).

```
#include <carConf.H>
```

Collaboration diagram for carConf:



### Public Member Functions

- [carConf](#) ([string](#) fileName)  
*default constructor, open config fileName and read*
- [carConf](#) ()  
*default constructor, open config "car.conf" and read*
- [~carConf](#) ()  
*default destructor*

### Public Attributes

- unsigned int [speedNeutralVal](#)  
*speed servo neutral position*
- unsigned int [speedMinVal](#)  
*speed servo minimum position*
- unsigned int [speedMaxVal](#)  
*speed servo maximum position*

- unsigned int [steerNeutralVal](#)  
*steering servo neutral position*
- unsigned int [steerMinVal](#)  
*steering servo minimum position*
- unsigned int [steerMaxVal](#)  
*steering servo maximum position*
- unsigned int [speedServoNum](#)  
*speed servo number on SSC*
- unsigned int [steerServoNum](#)  
*steering servo number on SSC*
- unsigned int [gearServoNum](#)  
*gear servo number on SSC*
- unsigned int [SSCbaud](#)  
*baud tranfer rate 2400 or 9600 only*
- [string SSCport](#)  
*SSC serial port device;.*

### 10.41.1 Detailed Description

This class uses [readConfig](#) to extract config parameters for [CarControl](#).

This class reads config parameters from "car.conf" or another specified config file and organizes them for use in [CarControl](#).

Definition at line 14 of file carConf.H.

### 10.41.2 Constructor & Destructor Documentation

#### 10.41.2.1 carConf::carConf ([string fileName](#))

default constructor, open config fileName and read

Definition at line 13 of file carConf.C.

References [readConfig::openFile\(\)](#).

#### 10.41.2.2 `carConf::carConf()`

default constructor, open config "car.conf" and read

Definition at line 7 of file carConf.C.

References `readConfig::openFile()`.

#### 10.41.2.3 `carConf::~~carConf()`

default destructor

Definition at line 19 of file carConf.C.

### 10.41.3 Member Data Documentation

#### 10.41.3.1 `unsigned int carConf::gearServoNum`

gear servo number on [SSC](#)

Definition at line 43 of file carConf.H.

Referenced by `CarControl::CarControl()`.

#### 10.41.3.2 `unsigned int carConf::speedMaxVal`

speed servo maximum position

Definition at line 31 of file carConf.H.

Referenced by `CarControl::CarControl()`.

#### 10.41.3.3 `unsigned int carConf::speedMinVal`

speed servo minimum position

Definition at line 29 of file carConf.H.

Referenced by `CarControl::CarControl()`.

#### 10.41.3.4 `unsigned int carConf::speedNeutralVal`

speed servo neutral position

Definition at line 27 of file carConf.H.

Referenced by `CarControl::CarControl()`.

**10.41.3.5 unsigned int [carConf::speedServoNum](#)**

speed servo number on [SSC](#)

Definition at line 39 of file carConf.H.

Referenced by CarControl::CarControl().

**10.41.3.6 unsigned int [carConf::SSCbaud](#)**

baud tranfer rate 2400 or 9600 only

Definition at line 45 of file carConf.H.

**10.41.3.7 string [carConf::SSCport](#)**

[SSC](#) serial port device;.

Definition at line 47 of file carConf.H.

**10.41.3.8 unsigned int [carConf::steerMaxVal](#)**

steering servo maximum position

Definition at line 37 of file carConf.H.

**10.41.3.9 unsigned int [carConf::steerMinVal](#)**

steering servo minimum position

Definition at line 35 of file carConf.H.

**10.41.3.10 unsigned int [carConf::steerNeutralVal](#)**

steering servo neutral position

Definition at line 33 of file carConf.H.

**10.41.3.11 unsigned int [carConf::steerServoNum](#)**

steering servo number on [SSC](#)

Definition at line 41 of file carConf.H.

Referenced by CarControl::CarControl().

The documentation for this class was generated from the following files:

- [carConf.H](#)
- [carConf.C](#)

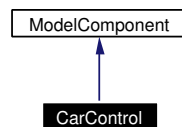


## 10.42 CarControl Class Reference

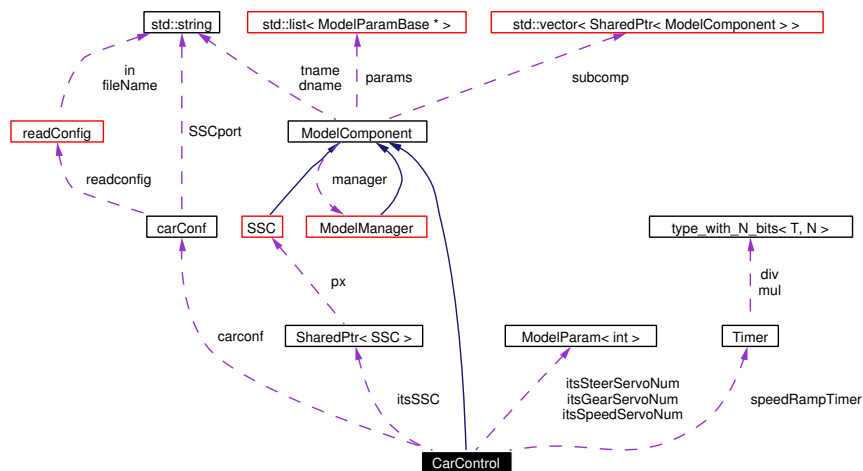
This class is an interface to pilot the RC car.

```
#include <carcontrol.H>
```

Inheritance diagram for CarControl:



Collaboration diagram for CarControl:



### Public Member Functions

- **CarControl** (**ModelManager** &mgr, const **std::string** &descrName="Car Controller", const **std::string** &tagName="CarControl")  
*General constructor.*
- **~CarControl** ()  
*Destructor.*
- void **calibrateSpeed** (const **byte** neutralval, const **byte** minval, const **byte** maxval)

*calibrate the speed SSC to new values*

- void [calibrateSteer](#) (const [byte](#) neutralval, const [byte](#) minval, const [byte](#) maxval)

*calibrate the steering SSC to new values*

- float [getSpeed](#) () const  
*gets speed value (-1.0 .. 1.0)*
- int [getGear](#) () const  
*gets gear value (0, 1, 2)*
- float [getSteering](#) () const  
*gets steering value (-1.0 .. 1.0)*
- bool [setSpeed](#) (const float newspeed)  
*sets speed value; input from -1.0 (full reverse) to 1.0 (full forward)*
- bool [setGear](#) (const int newgear)  
*set gear; input 0 (neutral) 1 (first gear, slow) or 2 (second gear, fast)*
- bool [setSteering](#) (const float newsteering)  
*sets steering angle; input from -1.0 (full left) to 1.0 (full right)*
- bool [toSpeedLinear](#) (const float newspeed, const int t)  
*linear ramp of speed from current speed to new speed*
- bool [toSpeedSigmoid](#) (const float newspeed, const int t)  
*sigmoid ramp of speed from current speed to new speed*
- void [rampSpeed](#) (const float newspeed, const int t, [SpeedRampType](#) behavior)  
*speed ramping function*
- void [abortRamp](#) (void)  
*aborts speed ramping*

## Protected Attributes

- [SharedPtr](#)< [SSC](#) > [itsSSC](#)  
*our servo controller*

- [ModelParam< int > itsSpeedServoNum](#)  
*servo number for speed*
- [ModelParam< int > itsSteerServoNum](#)  
*servo number for steering*
- [ModelParam< int > itsGearServoNum](#)  
*servo number for gear*

## Friends

- void \* [speedRampFunc](#) (void \*ptr)  
*routine for speed ramp thread*

### 10.42.1 Detailed Description

This class is an interface to pilot the RC car.

it will send orders to the servos

Definition at line 56 of file carcontrol.H.

### 10.42.2 Constructor & Destructor Documentation

#### 10.42.2.1 CarControl::CarControl ([ModelManager](#) & mgr, const [std::string](#) & descrName = "Car Controller", const [std::string](#) & tagName = "CarControl")

General constructor.

Definition at line 43 of file carcontrol.C.

References [ModelComponent::addSubComponent\(\)](#), [carConf::gearServoNum](#), [ModelParam< int >::getVal\(\)](#), [itsGearServoNum](#), [itsSpeedServoNum](#), [itsSteerServoNum](#), [ModelParam< int >::setVal\(\)](#), [carConf::speedMaxVal](#), [carConf::speedMinVal](#), [carConf::speedNeutralVal](#), [carConf::speedServoNum](#), and [carConf::steerServoNum](#).

#### 10.42.2.2 CarControl::~~CarControl ()

Destructor.

Definition at line 74 of file carcontrol.C.

### 10.42.3 Member Function Documentation

#### 10.42.3.1 void CarControl::abortRamp (void)

aborts speed ramping

Used to stop speed ramping thread so that main program has priority for setting speed.

Definition at line 212 of file carcontrol.C.

#### 10.42.3.2 void CarControl::calibrateSpeed (const **byte** *neutralval*, const **byte** *minval*, const **byte** *maxval*)

calibrate the speed **SSC** to new values

The speed **SSC** is calibrated to the values defined in a car.conf file during construction. This function allows the **SSC** to be recalibrated to new min/neutral/max values.

Definition at line 90 of file carcontrol.C.

References ModelParam< int >::getVal(), and itsSpeedServoNum.

#### 10.42.3.3 void CarControl::calibrateSteer (const **byte** *neutralval*, const **byte** *minval*, const **byte** *maxval*)

calibrate the steering **SSC** to new values

The steering **SSC** is calibrated to the values defined in a car.conf file during construction. This function allows the **SSC** to be recalibrated to new min/neutral/max values.

Definition at line 96 of file carcontrol.C.

References ModelParam< int >::getVal(), and itsSteerServoNum.

#### 10.42.3.4 int CarControl::getGear () const

gets gear value (0, 1, 2)

Definition at line 110 of file carcontrol.C.

References ModelParam< int >::getVal(), and itsGearServoNum.

#### 10.42.3.5 float CarControl::getSpeed () const

gets speed value (-1.0 .. 1.0)

Definition at line 102 of file carcontrol.C.

References ModelParam< int >::getVal(), and itsSpeedServoNum.

Referenced by `rampSpeed()`, `toSpeedLinear()`, and `toSpeedSigmoid()`.

#### 10.42.3.6 `float CarControl::getSteering () const`

gets steering value (-1.0 .. 1.0)

Definition at line 106 of file `carcontrol.C`.

References `ModelParam< int >::getVal()`, and `itsSteerServoNum`.

#### 10.42.3.7 `void CarControl::rampSpeed (const float newspeed, const int t, SpeedRampType behavior)`

speed ramping function

This function is used for gradually increasing the speed with a certain behavior over a certain period of time. The ramping will be done in a separate thread so this function will return immediately. It is not necessary to wait until the new speed has been reached before calling this function to modify the ramping parameters again. The ramping function will be stopped if [abortRamp\(\)](#) is called, or else if the thread continues other calls to [setSpeed\(\)](#) may seem to have no effect.

##### Parameters:

***newspeed*** new speed to be reached after ramping

***t*** time in which new speed will be reached (milliseconds)

***behavior*** specifies behavior of ramping function

Definition at line 188 of file `carcontrol.C`.

References `getSpeed()`, `Timer::reset()`, and `speedRampFunc`.

#### 10.42.3.8 `bool CarControl::setGear (const int newgear)`

set gear; input 0 (neutral) 1 (first gear, slow) or 2 (second gear, fast)

Definition at line 127 of file `carcontrol.C`.

References `ModelParam< int >::getVal()`, and `itsSpeedServoNum`.

#### 10.42.3.9 `bool CarControl::setSpeed (const float newspeed)`

sets speed value; input from -1.0 (full reverse) to 1.0 (full forward)

Definition at line 114 of file `carcontrol.C`.

References `ModelParam< int >::getVal()`, and `itsSpeedServoNum`.

Referenced by `toSpeedLinear()`, and `toSpeedSigmoid()`.

**10.42.3.10 bool CarControl::setSteering (const float *newsteering*)**

sets steering angle; input from -1.0 (full left) to 1.0 (full right)

Definition at line 123 of file carcontrol.C.

References ModelParam< int >::getVal(), and itsSteerServoNum.

**10.42.3.11 bool CarControl::toSpeedLinear (const float *newspeed*, const int *t*)**

linear ramp of speed from current speed to new speed

Speed will reach newspeed after t milliseconds

Definition at line 131 of file carcontrol.C.

References Timer::get(), getSpeed(), and setSpeed().

**10.42.3.12 bool CarControl::toSpeedSigmoid (const float *newspeed*, const int *t*)**

sigmoid ramp of speed from current speed to new speed

Speed will reach newspeed after t milliseconds

Definition at line 160 of file carcontrol.C.

References exp(), Timer::get(), getSpeed(), and setSpeed().

**10.42.4 Friends And Related Function Documentation****10.42.4.1 void\* speedRampFunc (void \* *ptr*) [friend]**

routine for speed ramp thread

Definition at line 222 of file carcontrol.C.

Referenced by rampSpeed().

**10.42.5 Member Data Documentation****10.42.5.1 [ModelParam<int>](#) [CarControl::itsGearServoNum](#) [protected]**

servo number for gear

Definition at line 132 of file carcontrol.H.

Referenced by CarControl(), and getGear().

**10.42.5.2** [ModelParam<int>](#) [CarControl::itsSpeedServoNum](#) [protected]

servo number for speed

Definition at line 130 of file carcontrol.H.

Referenced by [calibrateSpeed\(\)](#), [CarControl\(\)](#), [getSpeed\(\)](#), [setGear\(\)](#), and [setSpeed\(\)](#).

**10.42.5.3** [SharedPtr<SSC>](#) [CarControl::itsSSC](#) [protected]

our servo controller

Definition at line 129 of file carcontrol.H.

**10.42.5.4** [ModelParam<int>](#) [CarControl::itsSteerServoNum](#) [protected]

servo number for steering

Definition at line 131 of file carcontrol.H.

Referenced by [calibrateSteer\(\)](#), [CarControl\(\)](#), [getSteering\(\)](#), and [setSteering\(\)](#).

The documentation for this class was generated from the following files:

- [carcontrol.H](#)
- [carcontrol.C](#)

## 10.43 cascadeHold Class Reference

This is a container class for holding cascading energy values.

```
#include <cascadeHold.H>
```

### Public Attributes

- int [size](#)  
*vector size if used*
- int [iter](#)  
*Iteration.*
- int [b](#)  
*Angle of reciever.*
- int [a](#)  
*Angle of sender.*
- int [InspX](#)  
*position X of reciever*
- int [InspY](#)  
*position Y of reciever*
- float [cascadeStore](#)  
*Energy to send.*
- bool [pol](#)  
*polarity of other neruon*
- bool [sender](#)  
*sender of other neuron*
- bool [Mypol](#)  
*My polarity.*
- bool [Mysender](#)  
*My sender.*



### 10.43.1 Detailed Description

This is a container class for holding cascading energy values.

Definition at line 42 of file cascadeHold.H.

### 10.43.2 Member Data Documentation

#### 10.43.2.1 int cascadeHold::a

Angle of sender.

Definition at line 52 of file cascadeHold.H.

#### 10.43.2.2 int cascadeHold::b

Angle of reciever.

Definition at line 50 of file cascadeHold.H.

#### 10.43.2.3 float cascadeHold::cascadeStore

Energy to send.

Definition at line 58 of file cascadeHold.H.

#### 10.43.2.4 int cascadeHold::InspX

position X of reciever

Definition at line 54 of file cascadeHold.H.

#### 10.43.2.5 int cascadeHold::InspY

position Y of reciever

Definition at line 56 of file cascadeHold.H.

#### 10.43.2.6 int cascadeHold::iter

Iteration.

Definition at line 48 of file cascadeHold.H.

**10.43.2.7    bool [cascadeHold::Mypol](#)**

My polarity.

Definition at line 64 of file cascadeHold.H.

**10.43.2.8    bool [cascadeHold::Mysender](#)**

My sender.

Definition at line 66 of file cascadeHold.H.

**10.43.2.9    bool [cascadeHold::pol](#)**

polarity of other neruon

Definition at line 60 of file cascadeHold.H.

**10.43.2.10    bool [cascadeHold::sender](#)**

sender of other neuron

Definition at line 62 of file cascadeHold.H.

**10.43.2.11    int [cascadeHold::size](#)**

vector size if used

Definition at line 46 of file cascadeHold.H.

The documentation for this class was generated from the following file:

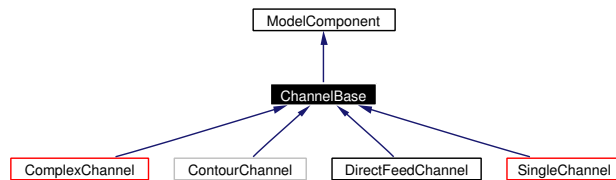
- [cascadeHold.H](#)

## 10.44 ChannelBase Class Reference

ChannelBase represents the various computational modules in [VisualCortex](#).

```
#include <ChannelBase.H>
```

Inheritance diagram for ChannelBase:



Collaboration diagram for ChannelBase:

### Public Member Functions

- [ChannelBase](#) ([ModelManager](#) &mgr, const [std::string](#) &descrName, const [std::string](#) &tagName, const [VisualFeature](#) vs)

*Default constructor. See [ModelComponent.H](#).*

- virtual [~ChannelBase](#) ()

*Virtual destructor ensures proper destruction of derived classes.*

- virtual void [exportOptions](#) (const int whatToExport, const bool recurse=true)

*Export some of our [ModelParam](#) parameters as command-line-options.*

### VisualFeature functions

- [VisualFeature](#) [visualFeature](#) () const  
*Get the visual feature.*
- void [setVisualFeature](#) (const [VisualFeature](#) v)  
*Set the visual feature.*
- bool [isHomogeneous](#) () const  
*Is the channel homogeneous?*
- void [setHomogeneous](#) (const bool h)  
*Set our homogeneous flag.*

### ParamMap functions

Subclasses should be sure to call *ChannelBase*'s version of these functions within their own implementations, so that *ChannelBase*'s parameters (such as its weights) are properly handled.

This interface is different from that provided by *ModelComponent*, as here we only save a subset of all of the *Channel*'s parameters, that are useful, e.g., for object recognition or feature-based attention biasing. Some of the values saved here may not even be *ModelParam* parameters.

- virtual void [readFrom](#) (const [ParamMap](#) &pmap)  
Read params from the [ParamMap](#).
- virtual void [writeTo](#) ([ParamMap](#) &pmap) const  
Write params to the [ParamMap](#).

### Weights+coeffs functions

These functions manage the weights and coefficients maintained by the channel. In general, the coefficients determine the relative weighting of any submaps that are maintained by the channel, while the weight determines the overall level of output of the channel. Thus, the coefficients are for controlling intra-channel competition, while the weights are for controlling inter-channel competition.

- void [setTotalWeight](#) (const double wt)  
Set the total weight allocated to this channel.
- double [getTotalWeight](#) () const  
return the value set for total weight
- void [setWeightPerChan](#) (const double wt)  
Set the weight allocated per subchannel.
- void [clampWeight](#) (const double cmin, const double cmax)  
Clamp the per-subchannel weight to the given range.
- void [cleanWeight](#) (const double eps)  
If the per-subchannel weight is less than eps, then it is set to zero.
- double [weightPerChan](#) () const  
Returns the weight allocated per subchannel.
- bool [isZeroWeight](#) () const  
Check if the channel's weight is zero.
- virtual void [cleanCoeffs](#) (const double eps)=0  
Set any coefficients less than eps to zero.

- virtual void [clampCoeffs](#) (const double cmin, const double cmax)=0  
*Clamp the weighting coefficients to the specified range.*
- virtual double [absSumCoeffs](#) () const=0  
*Return the sum of the absolute values of the weighting coefficients.*
- virtual void [normalizeCoeffs](#) (const double div)=0  
*Normalize the weighting coefficients by dividing by div.*
- virtual void [learn](#) (const [Image](#)< [byte](#) > &dmap, const double eta, const bool softmax=false, const int inthresh=20, const int outthresh=255)=0  
*Adjust the weighting coefficients according to the target map.*

### Input functions

There are several overloaded `input()` functions. These are provided to allow maximum efficiency, so that only the minimum required information must be passed to the channel. These functions delegate to the virtual function `doInput()`, passing null pointers in place of missing inputs. The size of the input image(s) can later be queried with `getInputDims()`. The timestamp parameter (`t`, in seconds) is used to compute transient responses to flickering stimuli.

- void [input](#) (const [Image](#)< float > &bwing, const double t, const [Image](#)< [byte](#) > &clipMask=[Image](#)< [byte](#) >())  
*Send monochrome (luminance) input to the channel.*
- void [input](#) (const [Image](#)< [PixRGB](#)< float > > &colimg, const double t, const [Image](#)< [byte](#) > &clipMask=[Image](#)< [byte](#) >())  
*Send multispectral (color) input to the channel.*
- void [input](#) (const [Image](#)< [PixRGB](#)< float > > &colimg, const [Image](#)< float > &bwing, const double t, const [Image](#)< [byte](#) > &clipMask=[Image](#)< [byte](#) >())  
*Send luminance and color input to the channel.*

### Access functions

- void [setOutputRange](#) (const float mini, const float maxi)  
*Set a fixed range of values for the raw output.*
- virtual bool [outputAvailable](#) () const=0  
*Query whether the channel is ready to give yield valid output.*
- virtual const [Dims](#) & [getMapDims](#) () const=0

*Return the dimensions of the output image.*

- virtual const **Dims** & **getInputDims** () const=0  
*Return the dimensions of the most recent input image.*
- virtual void **fillJet** (const **Point2D** &loc, **Jet**< float > &j, **std::vector**< int > &index) const=0  
*Fill the components of the given **Jet** that relate to this channel.*
- virtual void **fillJetNI** (const **Point2D** &loc, **Jet**< float > &j, **std::vector**< int > &index) const=0  
*Fill the components of the given **Jet** that relate to this channel.*
- virtual void **fillJetSpec** (**JetSpec** &js) const=0  
*Fill the specification of a **Jet** with data that relates to this channel.*
- virtual **uint** **numSubmaps** () const=0  
*Return the number of feature maps represented by this channel.*
- virtual **Image**< float > **getSubmap** (const **uint** index) const=0  
*Get the feature map at the given index.*
- virtual const char \* **getSubmapName** (const **uint** index) const=0  
*Get the name/description of the feature map at the given index.*
- virtual void **biasFeatures** (const double \*mean, const double \*sigma, int &idx)=0
- virtual void **getFeatures** (const **Point2D** &locn, **std::vector**< double > &mean) const=0
- virtual void **findWeight** ()=0  
*Weight of the channel is the maximum weight of its subchannels.*
- virtual bool **reco** (const **ParamMap** &pmap, double &mismatch)=0  
*Recognise this location in the channel by checking if it is obj.*
- virtual void **findLikelihood** (const **ParamMap** &pmap, double &likelihood)=0  
*what is the probability that the fixation belongs to this object?*
- virtual **Image**< float > **getOutput** ()=0  
*The output will be the size of the feature map level in the pyramid.*
- virtual void **drawResults** (**Image**< **PixRGB**< byte > > &traj)  
*Draws something channel-specific in the trajectory image.*
- virtual void **saveResults** (const **SharedPtr**< **OutputFrameSeries** > &ofs)  
*Save our various maps using an **OutputFrameSeries**.*

- virtual void [setListener](#) ([SharedPtr](#)< [ChannelListener](#) > listener)=0  
*Set a [ChannelListener](#) for this channel.*
- virtual [uint](#) [numChans](#) () const=0  
*number of subchannels for [ComplexChannel](#), 0 for [SingleChannel](#)*

### Protected Member Functions

- virtual void [doInput](#) (const [Image](#)< [PixRGB](#)< float > > \*colimg, const [Image](#)< float > \*bwimg, const double t, const [Image](#)< [byte](#) > &clipMask)=0  
*Subclasses override this implementation of the [input\(\)](#) function.*
- virtual void [killCaches](#) ()  
*Subclasses override this to clear any caches they may maintain.*

### Protected Attributes

- [ModelParam](#)< float > [itsOutputRangeMin](#)
- [ModelParam](#)< float > [itsOutputRangeMax](#)

#### 10.44.1 Detailed Description

ChannelBase represents the various computational modules in [VisualCortex](#).

Definition at line 79 of file ChannelBase.H.

#### 10.44.2 Constructor & Destructor Documentation

##### 10.44.2.1 ChannelBase::ChannelBase ([ModelManager](#) & mgr, const [std::string](#) & descrName, const [std::string](#) & tagName, const [VisualFeature](#) vs)

Default constructor. See [ModelComponent.H](#).

##### Parameters:

- mgr* our [ModelManager](#) (see [ModelManager.H](#))
- descrName* descriptive name for human useag
- tagName* name for [ParamMap](#) usage
- vs* The [VisualFeature](#) implemented by the channel

Definition at line 63 of file ChannelBase.C.

References MAXNORMMAX, MAXNORMMIN, and VisualFeature.

#### 10.44.2.2 ChannelBase::~~ChannelBase () [virtual]

Virtual destructor ensures proper destruction of derived classes.

Definition at line 74 of file ChannelBase.C.

### 10.44.3 Member Function Documentation

#### 10.44.3.1 virtual double ChannelBase::absSumCoeffs () const [pure virtual]

Return the sum of the absolute values of the weighting coefficients.

Implemented in [ComplexChannel](#), [DirectFeedChannel](#), and [SingleChannel](#).

Referenced by [ComplexChannel::absSumCoeffs\(\)](#), and [VisualCortex::normalizeParams\(\)](#).

#### 10.44.3.2 virtual void ChannelBase::biasFeatures (const double \* *mean*, const double \* *sigma*, int & *idx*) [pure virtual]

Bias the features at submap idx with mean, intra-class std. dev and inter-sub-class std. dev

Implemented in [ComplexChannel](#), [DirectFeedChannel](#), and [SingleChannel](#).

Referenced by [VisualCortex::biasFeatures\(\)](#), and [ComplexChannel::biasFeatures\(\)](#).

#### 10.44.3.3 virtual void ChannelBase::clampCoeffs (const double *cmin*, const double *cmax*) [pure virtual]

Clamp the weighting coefficients to the specified range.

Implemented in [ComplexChannel](#), [DirectFeedChannel](#), and [SingleChannel](#).

Referenced by [ComplexChannel::clampCoeffs\(\)](#), and [VisualCortex::clampParams\(\)](#).

#### 10.44.3.4 void ChannelBase::clampWeight (const double *cmin*, const double *cmax*)

Clamp the per-subchannel weight to the given range.

Definition at line 168 of file ChannelBase.C.



References `CLAMP()`, `killCaches()`, `setWeightPerChan()`, and `weightPerChan()`.

Referenced by `VisualCortex::clampParams()`.

#### 10.44.3.5 **virtual void ChannelBase::cleanCoeffs (const double *eps*)** [pure virtual]

Set any coefficients less than *eps* to zero.

Implemented in [ComplexChannel](#), [DirectFeedChannel](#), and [SingleChannel](#).

Referenced by `ComplexChannel::cleanCoeffs()`, and `VisualCortex::cleanParams()`.

#### 10.44.3.6 **void ChannelBase::cleanWeight (const double *eps*)**

If the per-subchannel weight is less than *eps*, then it is set to zero.

Definition at line 175 of file `ChannelBase.C`.

References `killCaches()`, `ModelParam< double >::setVal()`, and `weightPerChan()`.

Referenced by `VisualCortex::cleanParams()`.

#### 10.44.3.7 **virtual void ChannelBase::doInput (const [Image](#)< [PixRGB](#)< float > > \* *colimg*, const [Image](#)< float > \* *bwimg*, const double *t*, const [Image](#)< byte > & *clipMask*)** [protected, pure virtual]

Subclasses override this implementation of the `input()` function.

This will be called by one of the `input()` methods. For efficiency reasons (e.g. network bandwidth in parallel code), it is possible that the color or luminance input may be null, although the code should be wired up properly so that given subclasses do not receive a null input that they in fact need. Subclasses should be sure to ASSERT that any used inputs are non-null.

Implemented in [RGBConvolveChannel](#), [FlickerChannel](#), [DirectionChannel](#), [JunctionChannel](#), [EntropyChannel](#), [VarianceChannel](#), [TcorrChannel](#), [ScorrChannel](#), [InformationChannel](#), [DummyChannel](#), [PN03contrastChannel](#), [DisparityChannel](#), [HueChannel](#), [MultiConvolveChannel](#), [CompositeColorChannel](#), [ColorChannel](#), [SOColorChannel](#), [OrientationChannel](#), [MultiColorBandChannel](#), [MotionChannel](#), [StereoChannel](#), [LJunctionChannel](#), [TJunctionChannel](#), [XJunctionChannel](#), [EndPointChannel](#), [IntensityChannelBeo](#), [RedGreenChannelBeo](#), [BlueYellowChannelBeo](#), [GaborChannelBeo](#), [FlickerChannelBeo](#), [ColorBandChannelBeo](#), [DirectionChannelBeo](#), [DirectFeedChannel](#), [MultiDirectFeedChannel](#), [SingleChannel](#), and [SoxChannel](#).

Referenced by `input()`.

#### 10.44.3.8 **void ChannelBase::drawResults** (**Image**< **PixRGB**< **byte** > > & *traj*) [virtual]

Draws something channel-specific in the trajectory image.

The default implementation is a no-op; subclasses can override to do something interesting (see for example [TemplateMatchChannel](#)).

Reimplemented in [TemplateMatchChannel](#), and [ComplexChannel](#).

Definition at line 197 of file ChannelBase.C.

Referenced by [VisualCortex::drawResults\(\)](#), and [ComplexChannel::drawResults\(\)](#).

#### 10.44.3.9 **void ChannelBase::exportOptions** (const int *whatToExport*, const bool *recurse* = true) [virtual]

Export some of our [ModelParam](#) parameters as command-line-options.

See the base function in [ModelComponent.H](#) for info.

Reimplemented from [ModelComponent](#).

Reimplemented in [GaborChannel](#), [DirectionChannel](#), [JunctionChannel](#), [TcorrChannel](#), [ScorrChannel](#), [DummyChannel](#), [DisparityChannel](#), [ColorChannel](#), [OrientationChannel](#), [MultiColorBandChannel](#), [MotionChannel](#), [StereoChannel](#), [LJunctionChannel](#), [TJunctionChannel](#), [XJunctionChannel](#), [EndPointChannel](#), [ComplexChannel](#), [DirectFeedChannel](#), [SingleChannel](#), and [SoxChannel](#).

Definition at line 78 of file ChannelBase.C.

References [ModelComponent::exportOptions\(\)](#), [OPTEXP\\_CORE](#), and [ModelManager::requestOption\(\)](#).

Referenced by [SingleChannel::exportOptions\(\)](#), [DirectFeedChannel::exportOptions\(\)](#), and [ComplexChannel::exportOptions\(\)](#).

#### 10.44.3.10 **virtual void ChannelBase::fillJet** (const [Point2D](#) & *loc*, [Jet](#)< float > & *j*, [std::vector](#)< int > & *index*) const [pure virtual]

Fill the components of the given [Jet](#) that relate to this channel.

All relevant components of the [Jet](#) are set to this channel's values at the specified location across all pyramid levels. The channel will store its data starting at the given index in the jet, and will return the number of entries filled. The index vector will be modified as the various [Jet](#) components are explored, but will be restored to its original state upon completion.

Implemented in [ComplexChannel](#), [DirectFeedChannel](#), and [SingleChannel](#).

Referenced by [VisualCortex::fillJet\(\)](#), and [ComplexChannel::fillJet\(\)](#).

**10.44.3.11** `virtual void ChannelBase::fillJetNI (const Point2D & loc, Jet< float > & j, std::vector< int > & index) const` [pure virtual]

Fill the components of the given [Jet](#) that relate to this channel.

This version does not attempt to use interpolation

Implemented in [ComplexChannel](#), [DirectFeedChannel](#), and [SingleChannel](#).

Referenced by [VisualCortex::fillJetNI\(\)](#), and [ComplexChannel::fillJetNI\(\)](#).

**10.44.3.12** `virtual void ChannelBase::fillJetSpec (JetSpec & js) const` [pure virtual]

Fill the specification of a [Jet](#) with data that relates to this channel.

Implemented in [ComplexChannel](#), [DirectFeedChannel](#), and [SingleChannel](#).

Referenced by [ComplexChannel::fillJetSpec\(\)](#), and [VisualCortex::getJetSpec\(\)](#).

**10.44.3.13** `virtual void ChannelBase::findLikelihood (const ParamMap & pmap, double & likelihood) [pure virtual]`

what is the probability that the fixation belongs to this object?

Implemented in [ComplexChannel](#), [DirectFeedChannel](#), and [SingleChannel](#).

Referenced by [VisualCortex::findLikelihood\(\)](#), and [ComplexChannel::findLikelihood\(\)](#).

**10.44.3.14** `virtual void ChannelBase::findWeight ()` [pure virtual]

Weight of the channel is the maximum weight of its subchannels.

used for biasing to find targets

Implemented in [ComplexChannel](#), [DirectFeedChannel](#), and [SingleChannel](#).

Referenced by [VisualCortex::findWeight\(\)](#), and [ComplexChannel::findWeight\(\)](#).

**10.44.3.15** `virtual const Dims& ChannelBase::getInputDims () const` [pure virtual]

Return the dimensions of the most recent input image.

Implemented in [ComplexChannel](#), [DirectFeedChannel](#), and [SingleChannel](#).

Referenced by [ComplexChannel::getInputDims\(\)](#).

**10.44.3.16** `virtual const Dims& ChannelBase::getMapDims () const` [pure virtual]

Return the dimensions of the output image.

Implemented in [DummyChannel](#), [ComplexChannel](#), [DirectFeedChannel](#), and [SingleChannel](#).

Referenced by [ComplexChannel::getMapDims\(\)](#).

**10.44.3.17** `virtual Image<float> ChannelBase::getOutput ()` [pure virtual]

The output will be the size of the feature map level in the pyramid.

(I.e., the size of the output is given by [getMapDims\(\)](#)).

Implemented in [FlickerChannel](#), [EntropyChannel](#), [VarianceChannel](#), [TcorrChannel](#), [ScorrChannel](#), [InformationChannel](#), [PN03contrastChannel](#), [OrientationChannel](#), [ComplexChannel](#), [DirectFeedChannel](#), and [SingleChannel](#).

Referenced by [ComplexChannel::combineOutputs\(\)](#), and [VisualCortex::getOutput\(\)](#).

**10.44.3.18** `virtual Image<float> ChannelBase::getSubmap (const uint index) const` [pure virtual]

Get the feature map at the given index.

This function is intended to be implemented by subclasses so that it recurses into any nested channels, mapping nested submaps into a linear sequence of indices.

Implemented in [ComplexChannel](#), [DirectFeedChannel](#), and [SingleChannel](#).

Referenced by [ComplexChannel::getSubmap\(\)](#).

**10.44.3.19** `virtual const char* ChannelBase::getSubmapName (const uint index) const` [pure virtual]

Get the name/description of the feature map at the given index.

This function is intended to be implemented by subclasses so that it recurses into any nested channels, mapping nested submaps into a linear sequence of indices. The output of this function is intended only to be human-readable (e.g. for labeling image displays), and does not have to follow any particular parseable syntax.

Implemented in [HueChannel](#), [ComplexChannel](#), [DirectFeedChannel](#), and [SingleChannel](#).

Referenced by [ComplexChannel::getSubmapName\(\)](#).

**10.44.3.20 double ChannelBase::getTotalWeight () const**

return the value set for total weight

Definition at line 157 of file ChannelBase.C.

References `ModelParam< double >::getVal()`.

Referenced by `ComplexChannel::absSumCoeffs()`, `ComplexChannel::combineOutputs()`, `VisualCortex::findWeight()`, `ComplexChannel::findWeight()`, `VisualCortex::normalizeParams()`, `readFrom()`, and `writeTo()`.

**10.44.3.21 void ChannelBase::input (const Image< PixRGB< float > > & colimg, const Image< float > & bwimg, const double t, const Image< byte > & clipMask = Image< byte >())**

Send luminance and color input to the channel.

Definition at line 141 of file ChannelBase.C.

References `doInput()`, and `killCaches()`.

**10.44.3.22 void ChannelBase::input (const Image< PixRGB< float > > & colimg, const double t, const Image< byte > & clipMask = Image< byte >())**

Send multispectral (color) input to the channel.

Definition at line 133 of file ChannelBase.C.

References `doInput()`, and `killCaches()`.

**10.44.3.23 void ChannelBase::input (const Image< float > & bwimg, const double t, const Image< byte > & clipMask = Image< byte >())**

Send monochrome (luminance) input to the channel.

Definition at line 125 of file ChannelBase.C.

References `doInput()`, and `killCaches()`.

Referenced by `SoxChannel::doInput()`, `MultiDirectFeedChannel::doInput()`, `EndPointChannel::doInput()`, `XJunctionChannel::doInput()`, `TJunctionChannel::doInput()`, `LJunctionChannel::doInput()`, `MotionChannel::doInput()`, `MultiColorBandChannel::doInput()`, `OrientationChannel::doInput()`, `SOCColorChannel::doInput()`, `ColorChannel::doInput()`, `CompositeColorChannel::doInput()`, `MultiConvolveChannel::doInput()`, and `VisualCortex::input()`.

**10.44.3.24 bool ChannelBase::isHomogeneous () const**

Is the channel homogeneous?

A homogeneous channel is one whose subchannels (and their subs) have identical VisualFeatures, and those are also identical to our VisualFeature

Definition at line 99 of file ChannelBase.C.

References `ModelParam< bool >::getVal()`.

Referenced by `ComplexChannel::fillJet()`, `ComplexChannel::fillJetNI()`, and `ComplexChannel::fillJetSpec()`.

**10.44.3.25 bool ChannelBase::isZeroWeight () const**

Check if the channel's weight is zero.

If true, then the user of the channel may find it more efficient to simply ignore the output from this channel.

Definition at line 186 of file ChannelBase.C.

References `ModelParam< double >::getVal()`.

Referenced by `VisualCortex::getOutput()`.

**10.44.3.26 void ChannelBase::killCaches () [protected, virtual]**

Subclasses override this to clear any caches they may maintain.

**Subclasses must be sure to explicitly call their base class's version of this function!**

This function will be called by ChannelBase when a new input image is received, or when new parameters are read from a [ParamMap](#), etc. Subclasses may also call this function when they know that caches may have become invalid.

Reimplemented in [ComplexChannel](#), [DirectFeedChannel](#), [SingleChannel](#), and [SoxChannel](#).

Definition at line 210 of file ChannelBase.C.

Referenced by `clampWeight()`, `cleanWeight()`, `input()`, `SingleChannel::killCaches()`, `ComplexChannel::killCaches()`, `readFrom()`, `setTotalWeight()`, and `setWeightPerChan()`.

**10.44.3.27 virtual void ChannelBase::learn (const [Image](#)< byte > & dmap, const double eta, const bool softmax = false, const int inthresh = 20, const int outthresh = 255) [pure virtual]**

Adjust the weighting coefficients according to the target map.

Implemented in [ComplexChannel](#), [DirectFeedChannel](#), and [SingleChannel](#).

Referenced by [VisualCortex::learn\(\)](#), and [ComplexChannel::learn\(\)](#).

#### 10.44.3.28 **virtual void ChannelBase::normalizeCoeffs (const double div)** [pure virtual]

Normalize the weighting coefficients by dividing by div.

Implemented in [ComplexChannel](#), [DirectFeedChannel](#), and [SingleChannel](#).

Referenced by [ComplexChannel::normalizeCoeffs\(\)](#), and [VisualCortex::normalizeParams\(\)](#).

#### 10.44.3.29 **virtual uint ChannelBase::numChans () const** [pure virtual]

number of subchannels for [ComplexChannel](#), 0 for [SingleChannel](#)

Implemented in [ComplexChannel](#), [DirectFeedChannel](#), and [SingleChannel](#).

Referenced by [ShapeEstimator::getSingleChannel\(\)](#).

#### 10.44.3.30 **virtual uint ChannelBase::numSubmaps () const** [pure virtual]

Return the number of feature maps represented by this channel.

Implemented in [HueChannel](#), [ComplexChannel](#), [DirectFeedChannel](#), and [SingleChannel](#).

Referenced by [VisualCortex::normalizeParams\(\)](#), [ComplexChannel::numSubmaps\(\)](#), [VisualCortex::reco\(\)](#), [setWeightPerChan\(\)](#), and [weightPerChan\(\)](#).

#### 10.44.3.31 **virtual bool ChannelBase::outputAvailable () const** [pure virtual]

Query whether the channel is ready to give yield valid output.

In some cases (e.g. with motion channels), it may take several input frames before the channel is fully initialized.

Implemented in [EntropyChannel](#), [VarianceChannel](#), [TcorrChannel](#), [ScorrChannel](#), [InformationChannel](#), [DummyChannel](#), [PN03contrastChannel](#), [IntensityChannelBeo](#), [RedGreenChannelBeo](#), [BlueYellowChannelBeo](#), [GaborChannelBeo](#), [FlickerChannelBeo](#), [ColorBandChannelBeo](#), [DirectionChannelBeo](#), [ComplexChannel](#), [DirectFeedChannel](#), and [SingleChannel](#).

Referenced by [VisualCortex::getOutput\(\)](#), and [ComplexChannel::outputAvailable\(\)](#).

#### 10.44.3.32 **void ChannelBase::readFrom (const [ParamMap](#) & *pmap*)** [virtual]

Read params from the [ParamMap](#).

Reimplemented in [GaborChannel](#), [ComplexChannel](#), [DirectFeedChannel](#), and [SingleChannel](#).

Definition at line 107 of file ChannelBase.C.

References [getTotalWeight\(\)](#), [killCaches\(\)](#), [ParamMap::queryDoubleParam\(\)](#), and [setTotalWeight\(\)](#).

Referenced by [VisualCortex::readFrom\(\)](#), [SingleChannel::readFrom\(\)](#), [DirectFeedChannel::readFrom\(\)](#), and [ComplexChannel::readFrom\(\)](#).

#### 10.44.3.33 **virtual bool ChannelBase::reco (const [ParamMap](#) & *pmap*, double & *mismatch*)** [pure virtual]

Recognise this location in the channel by checking if it is obj.

Implemented in [ComplexChannel](#), [DirectFeedChannel](#), and [SingleChannel](#).

Referenced by [VisualCortex::reco\(\)](#), and [ComplexChannel::reco\(\)](#).

#### 10.44.3.34 **void ChannelBase::saveResults (const [SharedPtr](#)<[OutputFrameSeries](#) > & *ofs*)** [virtual]

Save our various maps using an [OutputFrameSeries](#).

The default implementation is a no-op; subclasses can override to do something interesting

Reimplemented in [ComplexChannel](#), and [SingleChannel](#).

Definition at line 204 of file ChannelBase.C.

Referenced by [VisualCortex::saveResults\(\)](#), and [ComplexChannel::saveResults\(\)](#).

#### 10.44.3.35 **void ChannelBase::setHomogeneous (const bool *h*)**

Set our homogeneous flag.

Definition at line 103 of file ChannelBase.C.

References [ModelParam< bool >::setVal\(\)](#).

Referenced by [ComplexChannel::addSubChan\(\)](#), [SoxChannel::buildSubChans\(\)](#), [EndPointChannelSurprise::buildSubChans\(\)](#), [XJunctionChannelSurprise::buildSubChans\(\)](#), [TJunctionChannelSurprise::buildSubChans\(\)](#), and [LJunctionChannel-](#)



Surprise::buildSubChans(), MotionChannelSurprise::buildSubChans(), MultiColorBandChannelSurprise::buildSubChans(), OrientationChannelSurprise::buildSubChans(), MotionChannelBeo::buildSubChans(), MultiColorBandChannelBeo::buildSubChans(), OrientationChannelBeo::buildSubChans(), StereoChannel::buildSubChans(), EndPointChannel::buildSubChans(), XJunctionChannel::buildSubChans(), TJunctionChannel::buildSubChans(), LJunctionChannel::buildSubChans(), MotionChannel::buildSubChans(), MultiColorBandChannel::buildSubChans(), OrientationChannel::buildSubChans(), ColorChannelSurprise::ColorChannelSurprise(), MultiConvolveChannelSurprise::MultiConvolveChannelSurprise(), and SOColorChannelSurprise::SOColorChannelSurprise().

#### 10.44.3.36 **virtual void ChannelBase::setListener** ([SharedPtr<ChannelListener](#) > *listener*) [pure virtual]

Set a [ChannelListener](#) for this channel.

The [ChannelListener](#) which will get called once for each center/surround pairing.

Implemented in [ComplexChannel](#), [DirectFeedChannel](#), and [SingleChannel](#).

Referenced by [ComplexChannel::setListener\(\)](#).

#### 10.44.3.37 **void ChannelBase::setOutputRange** (const float *mini*, const float *maxi*)

Set a fixed range of values for the raw output.

By default, the range is set to [MAXNORMMIN .. MAXNORMMAX] at construction. If non-zero values are provided here, in [getOutput\(\)](#) we will cumulate our various submaps, apply the provided range, apply spatial competition for salience, apply our total weight, and return the result. If a range [0.0 .. 0.0] is provided here, only spatial competition and weights will be applied, but the first step of applying the range will be skipped. This must be called before [start\(\)](#).

Definition at line 190 of file [ChannelBase.C](#).

References [ModelParam< float >::setVal\(\)](#).

Referenced by [SingleChannel::start1\(\)](#), and [ComplexChannel::start2\(\)](#).

#### 10.44.3.38 **void ChannelBase::setTotalWeight** (const double *wt*)

Set the total weight allocated to this channel.

This weight will be divided amongst any subchannels managed by this channel. So, for example, if the total weight is set to 2.0, and the channel has four subchannels, then each subchannel will get an effective weight of 0.5. The initial setting is 1.0 for most

channels. But, If you are handling b/w images, for instance, you may set the weight for the color channel to 0.0 and save computations.

Definition at line 150 of file ChannelBase.C.

References `killCaches()`, and `ModelParam< double >::setVal()`.

Referenced by `VisualCortexSurprise::addDefaultChannels()`, `VisualCortex-Beo::addDefaultChannels()`, `VisualCortex::addDefaultChannels()`, `CompositeColorChannel::CompositeColorChannel()`, `CompositeColorChannelSurprise::CompositeColorChannelSurprise()`, `VisualCortex::findWeight()`, `SingleChannel::findWeight()`, `ComplexChannel::findWeight()`, `readFrom()`, `SOCColorChannel::SOCColorChannel()`, and `SOCColorChannelSurprise::SOCColorChannelSurprise()`.

#### 10.44.3.39 void ChannelBase::setVisualFeature (const VisualFeature v)

Set the visual feature.

Definition at line 95 of file ChannelBase.C.

References `ModelParam< VisualFeature >::setVal()`.

Referenced by `JunctionChannel::JunctionChannel()`.

#### 10.44.3.40 void ChannelBase::setWeightPerChan (const double wt)

Set the weight allocated per subchannel.

The total weight will then be the per-channel weight times the number of subchannels.

Definition at line 161 of file ChannelBase.C.

References `killCaches()`, `numSubmaps()`, and `ModelParam< double >::setVal()`.

Referenced by `clampWeight()`.

#### 10.44.3.41 VisualFeature ChannelBase::visualFeature () const

Get the visual feature.

Definition at line 91 of file ChannelBase.C.

References `ModelParam< VisualFeature >::getVal()`, and `VisualFeature`.

Referenced by `ComplexChannel::addSubChan()`, `EndPointChannelSurprise::buildSubChans()`, `XJunctionChannelSurprise::buildSubChans()`, `TJunctionChannelSurprise::buildSubChans()`, `LJunctionChannelSurprise::buildSubChans()`, `EndPointChannel::buildSubChans()`, `XJunctionChannel::buildSubChans()`, `TJunctionChannel::buildSubChans()`, `LJunctionChannel::buildSubChans()`, `SingleChannel::fillJet()`, `SingleChannel::fillJetNI()`, `SingleChannel::fillJetSpec()`, `ComplexChannel::fill-`

JetSpec(), and SingleChannelBeo::sendInput().

#### 10.44.3.42 double ChannelBase::weightPerChan () const

Returns the weight allocated per subchannel.

Definition at line 182 of file ChannelBase.C.

References ModelParam< double >::getVal(), and numSubmaps().

Referenced by clampWeight(), and cleanWeight().

#### 10.44.3.43 void ChannelBase::writeTo (ParamMap & pmap) const [virtual]

Write params to the [ParamMap](#).

Reimplemented in [GaborChannel](#), [ComplexChannel](#), [DirectFeedChannel](#), and [SingleChannel](#).

Definition at line 119 of file ChannelBase.C.

References getTotalWeight(), and ParamMap::putDoubleParam().

Referenced by VisualCortex::writeTo(), SingleChannel::writeTo(), DirectFeedChannel::writeTo(), and ComplexChannel::writeTo().

The documentation for this class was generated from the following files:

- [ChannelBase.H](#)
- [ChannelBase.C](#)

## 10.45 ChannelListener Class Reference

A callback class for use with [ChannelBase](#) and its derivatives.

```
#include <ChannelBase.H>
```

### Public Member Functions

- virtual void [onEachCS](#) (const [ChannelBase](#) &channel, const [uint](#) lev, const [uint](#) del, const [Image](#)< float > &cs)=0

*Subclasses of [ChannelBase](#) use this interface to callback to listeners.*

#### 10.45.1 Detailed Description

A callback class for use with [ChannelBase](#) and its derivatives.

ChannelListener gets called back from Channel::getRawOutput() for each center/surround pairing.

Definition at line 63 of file ChannelBase.H.

#### 10.45.2 Member Function Documentation

**10.45.2.1** virtual void ChannelListener::onEachCS (const [ChannelBase](#) &channel, const [uint](#) lev, const [uint](#) del, const [Image](#)< float > &cs)  
[pure virtual]

Subclasses of [ChannelBase](#) use this interface to callback to listeners.

Subclasses should pass \*this as the first argument to [onEachCS\(\)](#), so that the client can figure out which channel triggered the callback.

The documentation for this class was generated from the following files:

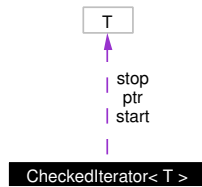
- [ChannelBase.H](#)
- [ChannelBase.C](#)

## 10.46 CheckedIterator< T > Class Template Reference

A range-checked iterator class for memory debugging.

```
#include <CheckedIterator.H>
```

Collaboration diagram for CheckedIterator< T >:



### Public Types

- typedef T [value\\_type](#)  
*This typedef is used by STL functions.*
- typedef T \* [pointer](#)  
*This typedef is used by STL functions.*
- typedef T & [reference](#)  
*This typedef is used by STL functions.*
- typedef ptrdiff\_t [difference\\_type](#)  
*This typedef is used by STL functions.*
- typedef std::random\_access\_iterator\_tag [iterator\\_category](#)  
*This typedef is used by STL functions.*

### Public Member Functions

- [CheckedIterator](#) ()  
*Default constructor.*
- [CheckedIterator](#) (T \*p, T \*a, T \*b)  
*Constructor.*

- [CheckedIterator](#) (T \*p, T \*b)  
*Constructor.*
- template<class UU> [CheckedIterator](#) (const [CheckedIterator](#)< UU > &other)  
*Constructor.*
- void **ck\_range** (const diff\_t offset) const

### Pointer arithmetic operators

- [CheckedIterator](#)< T > & **operator++** ()  
*Pre-increment operator.*
- [CheckedIterator](#)< T > **operator++** (int)  
*Post-increment operator.*
- [CheckedIterator](#)< T > & **operator--** ()  
*Pre-decrement operator.*
- [CheckedIterator](#)< T > **operator--** (int)  
*Post-decrement operator.*
- void **operator+=** (diff\_t d)
- void **operator-=** (diff\_t d)
- [CheckedIterator](#)< T > **operator+** (diff\_t d) const
- [CheckedIterator](#)< T > **operator-** (diff\_t d) const
- diff\_t **operator-** ([CheckedIterator](#)< T > other) const

### Comparison operators

- bool **operator==** (const [CheckedIterator](#)< T > &other) const
- bool **operator!=** (const [CheckedIterator](#)< T > &other) const
- bool **operator<** (const [CheckedIterator](#)< T > &other) const
- bool **operator>** (const [CheckedIterator](#)< T > &other) const
- bool **operator<=** (const [CheckedIterator](#)< T > &other) const
- bool **operator>=** (const [CheckedIterator](#)< T > &other) const

### Dereferencing operators

- T \* **operator** → () const
- T & **operator** \* () const
- T & **operator**[ ] (diff\_t d) const

## Public Attributes

- T \* [ptr](#)  
*Pointer to the data.*
- T \* [start](#)  
*Pointer to the first element we can iterate on.*
- T \* [stop](#)  
*Pointer to the last element we can iterate on.*

### 10.46.1 Detailed Description

**template<class T> class CheckedIterator< T >**

A range-checked iterator class for memory debugging.

A CheckedIterator<T> object exposes essentially the same interface as a T\*, but stores a start/stop range that should bound the iterator. This range is checked with a call to `ck_range()` every time the iterator is dereferenced, by either [operator\\*\(\)](#) or `operator → ()`.

Definition at line 55 of file CheckedIterator.H.

### 10.46.2 Member Typedef Documentation

**10.46.2.1** **template<class T> typedef ptrdiff\_t [CheckedIterator< T >::difference\\_type](#)**

This typedef is used by STL functions.

Definition at line 68 of file CheckedIterator.H.

**10.46.2.2** **template<class T> typedef std::random\_access\_iterator\_tag [CheckedIterator< T >::iterator\\_category](#)**

This typedef is used by STL functions.

Definition at line 71 of file CheckedIterator.H.

**10.46.2.3** **template<class T> typedef T\* [CheckedIterator< T >::pointer](#)**

This typedef is used by STL functions.

Definition at line 62 of file CheckedIterator.H.

**10.46.2.4    template<class T> typedef T& [CheckedIterator](#)< T >::reference**

This typedef is used by STL functions.

Definition at line 65 of file CheckedIterator.H.

**10.46.2.5    template<class T> typedef T [CheckedIterator](#)< T >::value\_type**

This typedef is used by STL functions.

Definition at line 59 of file CheckedIterator.H.

**10.46.3    Constructor & Destructor Documentation****10.46.3.1    template<class T> [CheckedIterator](#)< T >::CheckedIterator ()  
[inline]**

Default constructor.

This builds an iterator with an empty valid range, so a new value must be assigned to the iterator before it can be dereferenced. However, providing a default constructor helps keep source compatibility with non-debug iterators (i.e., raw pointers).

Definition at line 151 of file CheckedIterator.H.

Referenced by `CheckedIterator< T >::operator++()`, and `CheckedIterator< T >::operator--()`.

**10.46.3.2    template<class T> [CheckedIterator](#)< T >::CheckedIterator (T \* *p*,  
T \* *a*, T \* *b*) [inline]**

Constructor.

Definition at line 157 of file CheckedIterator.H.

**10.46.3.3    template<class T> [CheckedIterator](#)< T >::CheckedIterator (T \* *p*,  
T \* *b*) [inline]**

Constructor.

Definition at line 163 of file CheckedIterator.H.



**10.46.3.4** `template<class T> template<class UU> CheckedIterator< T  
>::CheckedIterator (const CheckedIterator< UU > & other)  
[inline]`

Constructor.

Definition at line 169 of file CheckedIterator.H.

## 10.46.4 Member Function Documentation

**10.46.4.1** `template<class T> CheckedIterator< T > CheckedIterator< T  
>::operator++ (int) [inline]`

Post-increment operator.

Definition at line 180 of file CheckedIterator.H.

References CheckedIterator< T >::CheckedIterator(), CheckedIterator< T >::ptr,  
CheckedIterator< T >::start, and CheckedIterator< T >::stop.

**10.46.4.2** `template<class T> CheckedIterator< T > & CheckedIterator< T  
>::operator++ () [inline]`

Pre-increment operator.

Definition at line 175 of file CheckedIterator.H.

References CheckedIterator< T >::ptr.

**10.46.4.3** `template<class T> CheckedIterator< T > CheckedIterator< T  
>::operator-- (int) [inline]`

Post-decrement operator.

Definition at line 190 of file CheckedIterator.H.

References CheckedIterator< T >::CheckedIterator(), CheckedIterator< T >::ptr,  
CheckedIterator< T >::start, and CheckedIterator< T >::stop.

**10.46.4.4** `template<class T> CheckedIterator< T > & CheckedIterator< T  
>::operator-- () [inline]`

Pre-decrement operator.

Definition at line 185 of file CheckedIterator.H.

References CheckedIterator< T >::ptr.

## 10.46.5 Member Data Documentation

### 10.46.5.1 `template<class T> T* CheckedIterator< T >::ptr`

Pointer to the data.

Definition at line 136 of file `CheckedIterator.H`.

Referenced by `CheckedIterator< T >::operator++()`, and `CheckedIterator< T >::operator--()`.

### 10.46.5.2 `template<class T> T* CheckedIterator< T >::start`

Pointer to the first element we can iterate on.

Definition at line 139 of file `CheckedIterator.H`.

Referenced by `CheckedIterator< T >::operator++()`, and `CheckedIterator< T >::operator--()`.

### 10.46.5.3 `template<class T> T* CheckedIterator< T >::stop`

Pointer to the last element we can iterate on.

Definition at line 142 of file `CheckedIterator.H`.

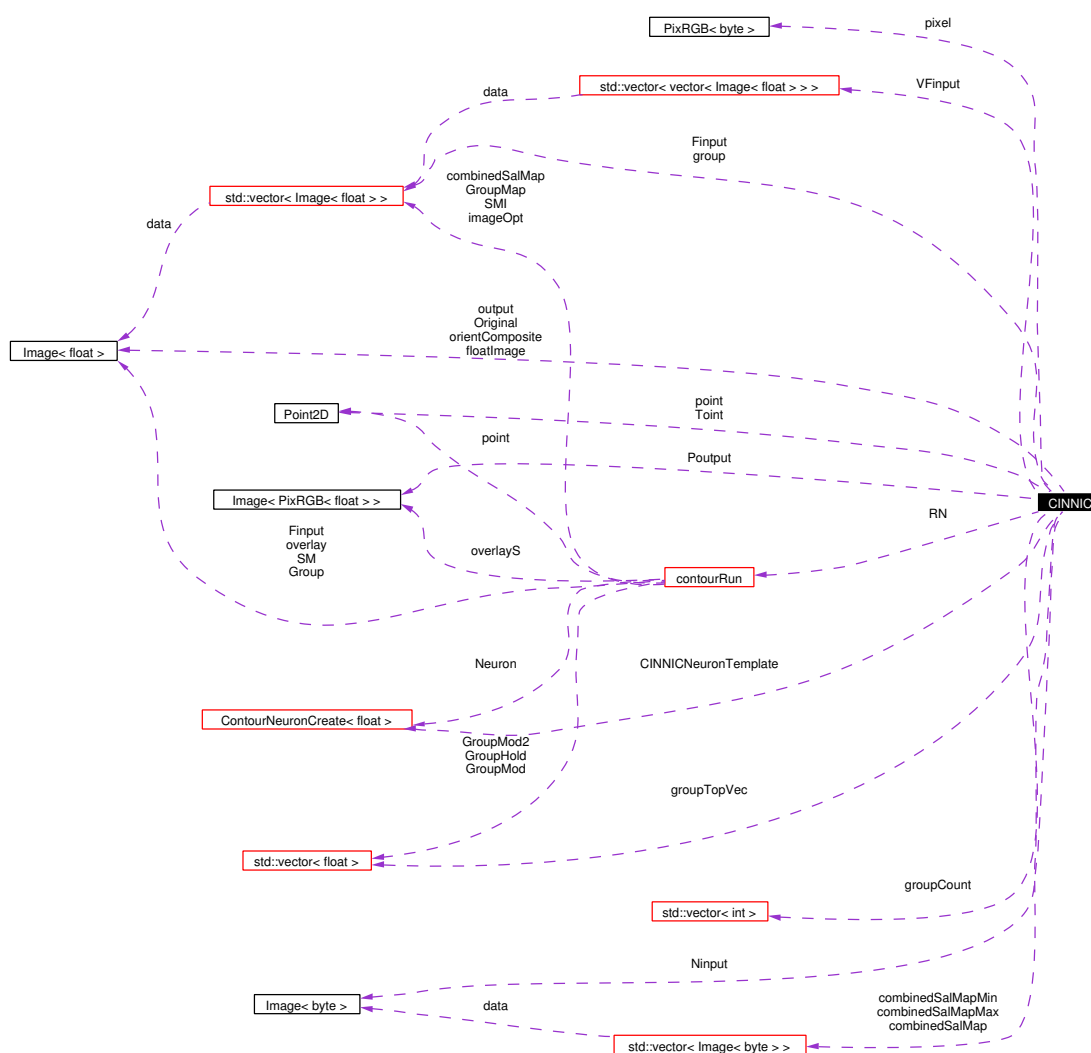
Referenced by `CheckedIterator< T >::operator++()`, and `CheckedIterator< T >::operator--()`.

The documentation for this class was generated from the following file:

- [CheckedIterator.H](#)

Main class to run an image in CINNICtest for contours.

Collaboration diagram for CINNIC:



## Public Member Functions

- void [viewNeuronTemplate](#) ([ContourNeuronCreate](#)< float > &NeuronTemplate, [readConfig](#) &config)

*View the output visually of the contour neuron templates.*

- void [RunSimpleImage](#) ([ContourNeuronCreate](#)< float > &NeuronTemplate, [Image](#)< byte > &input, [readConfig](#) &config)

*Run an image (this is the important method).*

- void [convolveTest](#) ([ContourNeuronCreate](#)< float > &NeuronTemplate, [readConfig](#) &config, [Image](#)< float > &testImage)

*test a simple image against itself to find the value of convolution*

## Public Attributes

- [Image](#)< byte > [Ninput](#)

*Aux [Image](#) holder.*

- char \* [filename](#)

*This is the file name as supplied by argv.*

- char \* [savefilename](#)

*this is what to save output as*

### 10.47.1 Detailed Description

Main class to run an image in CINNICtest for contours.

This is the central class at this point for running CINNIC. It works by creating a CINNIC template for neuron connections in [contourNeuron](#). It then imports an image as defines from the command line. The image is processed for orientations using the [orientedFilter](#) in [Image](#) as well as a few other processes. The image is then reduced and a 3D map of the image created and stored in [ImageMap](#). This map and the neuron are then fed into [contourRun](#) which runs the 3D image against the template neuron. Output is produced in many ways including several raw data dumps and image dumps.

Definition at line 68 of file CINNIC.H.

## 10.47.2 Member Function Documentation

### 10.47.2.1 void CINNIC::convolveTest ([ContourNeuronCreate](#)< float > & *NeuronTemplate*, [readConfig](#) & *config*, [Image](#)< float > & *testImage*)

test a simple image against itself to find the value of convolution

Definition at line 105 of file CINNIC.C.

References [PropHold::angABD](#), [ContourNeuronCreate](#)< TC2 >::FourDNeuralMap, [Image](#)< T >::getHeight(), [Image](#)< T >::getVal(), [Image](#)< T >::getWidth(), [Image](#)< T >::resize(), and [Image](#)< T >::setVal().

Referenced by [main\(\)](#).

### 10.47.2.2 void CINNIC::RunSimpleImage ([ContourNeuronCreate](#)< float > & *NeuronTemplate*, [Image](#)< byte > & *input*, [readConfig](#) & *config*)

Run an image (this is the important method).

Definition at line 248 of file CINNIC.C.

References [Timer::get\(\)](#), [Timer::reset\(\)](#), [Image](#)< float >::resize(), and [uint64](#).

Referenced by [main\(\)](#).

### 10.47.2.3 void CINNIC::viewNeuronTemplate ([ContourNeuronCreate](#)< float > & *NeuronTemplate*, [readConfig](#) & *config*)

View the output visually of the contour neuron templates.

Definition at line 55 of file CINNIC.C.

References [PropHold::angABD](#), [ContourNeuronCreate](#)< TC2 >::FourDNeuralMap, [Image](#)< float >::getMinMaxAvg(), [NO\\_INIT](#), [normalizeRGPolar\(\)](#), [Image](#)< T >::paste(), [Image](#)< T >::resize(), [Image](#)< float >::setVal(), and [Raster::Visu\(\)](#).

Referenced by [main\(\)](#).

## 10.47.3 Member Data Documentation

### 10.47.3.1 char\* [CINNIC::filename](#)

This is the file name as supplied by argv.

Definition at line 148 of file CINNIC.H.

Referenced by [main\(\)](#).

**10.47.3.2   [Image<byte>](#) [CINNIC::Ninput](#)**

Aux [Image](#) holder.

Definition at line 140 of file CINNIC.H.

Referenced by [main\(\)](#).

**10.47.3.3   [char\\*](#) [CINNIC::savefilename](#)**

this is what to save output as

Definition at line 150 of file CINNIC.H.

Referenced by [main\(\)](#).

The documentation for this class was generated from the following files:

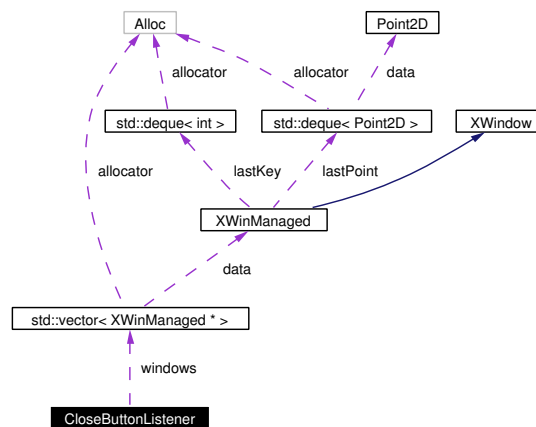
- [CINNIC.H](#)
- [CINNIC.C](#)

## 10.48 CloseButtonListener Class Reference

watch for close button actions with all registered [XWinManaged](#) windows

```
#include <XWinManaged.H>
```

Collaboration diagram for CloseButtonListener:



### Public Member Functions

- void [add](#) ([XWinManaged](#) \*win)  
*add a window to the list by pointer*
- void [add](#) ([XWinManaged](#) &win)  
*add a window to the list by reference*
- void [remove](#) ([XWinManaged](#) \*win)  
*remove a window from the list by pointer*
- void [remove](#) ([XWinManaged](#) &win)  
*remove a window from the list by reference*
- bool [pressedAnyCloseButton](#) (void)  
*has a close button been pressed at any of the windows?*
- void [waitForClose](#) ()  
*wait for a close button to be pressed*

### 10.48.1 Detailed Description

watch for close button actions with all registered [XWinManaged](#) windows

Definition at line 180 of file XWinManaged.H.

### 10.48.2 Member Function Documentation

#### 10.48.2.1 void CloseButtonListener::add ([XWinManaged](#) & win) [inline]

add a window to the list by reference

Definition at line 217 of file XWinManaged.H.

References add().

#### 10.48.2.2 void CloseButtonListener::add ([XWinManaged](#) \* win) [inline]

add a window to the list by pointer

Definition at line 214 of file XWinManaged.H.

Referenced by add(), and main().

#### 10.48.2.3 bool CloseButtonListener::pressedAnyCloseButton (void) [inline]

has a close button been pressed at any of the windows?

This function will seg fault if the list contains pointers to windows that don't exist anymore. Make sure that you remove windows from the CloseButtonListener before you explicitly delete them!

Definition at line 232 of file XWinManaged.H.

References uint.

Referenced by main(), and waitForClose().

#### 10.48.2.4 void CloseButtonListener::remove ([XWinManaged](#) & win) [inline]

remove a window from the list by reference

Definition at line 228 of file XWinManaged.H.

References remove().



**10.48.2.5 void CloseButtonListener::remove ([XWinManaged](#) \* win)**  
[inline]

remove a window from the list by pointer

Definition at line 221 of file XWinManaged.H.

Referenced by remove().

**10.48.2.6 void CloseButtonListener::waitForClose ()** [inline]

wait for a close button to be pressed

run in an idle loop until a close button is pressed

Definition at line 240 of file XWinManaged.H.

References pressedAnyCloseButton().

Referenced by main().

The documentation for this class was generated from the following file:

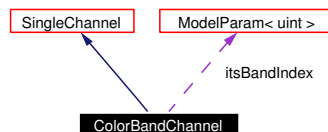
- [XWinManaged.H](#)

## 10.49 ColorBandChannel Class Reference

A color band channel.

```
#include <Channels.H>
```

Inheritance diagram for ColorBandChannel: Collaboration diagram for ColorBandChannel:



### Public Member Functions

- [ColorBandChannel](#) ([ModelManager](#) &mgr, const [uint](#) bandidx)  
*Construct with standard params.*
- virtual [~ColorBandChannel](#) ()  
*Destructor.*

### Protected Attributes

- [ModelParam](#)< [uint](#) > [itsBandIndex](#)

#### 10.49.1 Detailed Description

A color band channel.

Definition at line 318 of file Channels.H.

#### 10.49.2 Constructor & Destructor Documentation

##### 10.49.2.1 ColorBandChannel::ColorBandChannel ([ModelManager](#) & mgr, const [uint](#) bandidx)

Construct with standard params.

Definition at line 495 of file Channels.C.

References COLBAND, makeShared(), ModelComponent::setDescriptiveName(), ModelComponent::setTagName(), ModelParam< bool >::setVal(), and uint.

#### 10.49.2.2 ColorBandChannel::~ColorBandChannel () [virtual]

Destructor.

Definition at line 512 of file Channels.C.

The documentation for this class was generated from the following files:

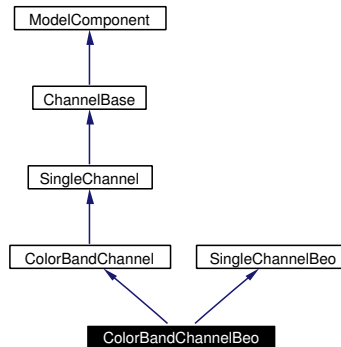
- [Channels.H](#)
- [Channels.C](#)

## 10.50 ColorBandChannelBeo Class Reference

Beowulf version of [ColorBandChannel](#).

```
#include <ChannelsBeo.H>
```

Inheritance diagram for ColorBandChannelBeo:



Collaboration diagram for ColorBandChannelBeo:

### Public Member Functions

- **ColorBandChannelBeo** ([ModelManager](#) &mgr, const [uint](#) bandidx, const [SharedPtr](#)< [Beowulf](#) > &beow)
- virtual bool [outputAvailable](#) () const  
*Query whether the channel is ready to give yield valid output.*

### Protected Member Functions

- virtual void [doInput](#) (const [Image](#)< [PixRGB](#)< float > > \*, const [Image](#)< float > > \*bwimg, const double t, const [Image](#)< [byte](#) > &clipMask)  
*SingleChannel requires only luminance input.*
- virtual void [start1](#) ()

#### 10.50.1 Detailed Description

Beowulf version of [ColorBandChannel](#).

Definition at line 128 of file ChannelsBeo.H.

## 10.50.2 Member Function Documentation

**10.50.2.1** `void ColorBandChannelBeo::doInput (const Image< PixRGB< float > > *, const Image< float > * bwimg, const double t, const Image< byte > & clipMask)` [protected, virtual]

[SingleChannel](#) requires only luminance input.

Reimplemented from [SingleChannel](#).

Definition at line 199 of file ChannelsBeo.C.

References [SingleChannelBeo::sendInput\(\)](#).

**10.50.2.2** `bool ColorBandChannelBeo::outputAvailable () const` [virtual]

Query whether the channel is ready to give yield valid output.

In some cases (e.g. with motion channels), it may take several input frames before the channel is fully initialized.

Reimplemented from [SingleChannel](#).

Definition at line 192 of file ChannelsBeo.C.

References [SingleChannel::outputAvailable\(\)](#).

**10.50.2.3** `void ColorBandChannelBeo::start1 ()` [protected, virtual]

If you overload this, make sure you call [SingleChannel::start1\(\)](#) at the beginning of your overload

Reimplemented from [SingleChannel](#).

Definition at line 203 of file ChannelsBeo.C.

References [SingleChannelBeo::requestBeowulfNode\(\)](#), and [SingleChannel::start1\(\)](#).

The documentation for this class was generated from the following files:

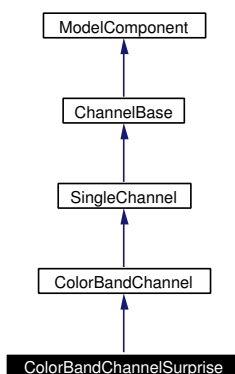
- [ChannelsBeo.H](#)
- [ChannelsBeo.C](#)

## 10.51 ColorBandChannelSurprise Class Reference

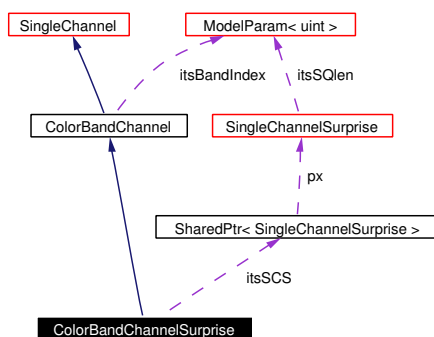
A color band channel.

```
#include <ChannelsSurprise.H>
```

Inheritance diagram for ColorBandChannelSurprise:



Collaboration diagram for ColorBandChannelSurprise:



### Public Member Functions

- **ColorBandChannelSurprise** (**ModelManager** &mgr, const **uint** bandidx)  
*Construct with standard params.*
- virtual **~ColorBandChannelSurprise** ()  
*Destructor.*

- virtual [Image](#)< float > [computeSubmap](#) (const [uint](#) index) const  
*[computeSubmap\(\)](#) is special and will return surprise*

## Protected Attributes

- [SharedPtr](#)< [SingleChannelSurprise](#) > itsSCS  
*our surprise computer*

### 10.51.1 Detailed Description

A color band channel.

Definition at line 304 of file ChannelsSurprise.H.

### 10.51.2 Constructor & Destructor Documentation

#### 10.51.2.1 ColorBandChannelSurprise::ColorBandChannelSurprise ([ModelManager](#) & mgr, const [uint](#) bandidx)

Construct with standard params.

Definition at line 276 of file ChannelsSurprise.C.

References [ModelComponent::addSubComponent\(\)](#), and [uint](#).

#### 10.51.2.2 ColorBandChannelSurprise::~~ColorBandChannelSurprise () [virtual]

Destructor.

Definition at line 283 of file ChannelsSurprise.C.

### 10.51.3 Member Function Documentation

#### 10.51.3.1 [Image](#)< float > ColorBandChannelSurprise::computeSubmap (const [uint](#) index) const [virtual]

[computeSubmap\(\)](#) is special and will return surprise

Reimplemented from [SingleChannel](#).

Definition at line 286 of file ChannelsSurprise.C.

References uint.

#### 10.51.4 Member Data Documentation

##### 10.51.4.1 [SharedPtr<SingleChannelSurprise> ColorBandChannelSurprise::itsSCS](#) [protected]

our surprise computer

Definition at line 317 of file ChannelsSurprise.H.

The documentation for this class was generated from the following files:

- [ChannelsSurprise.H](#)
- ChannelsSurprise.C

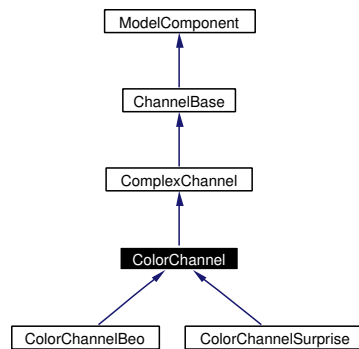


## 10.52 ColorChannel Class Reference

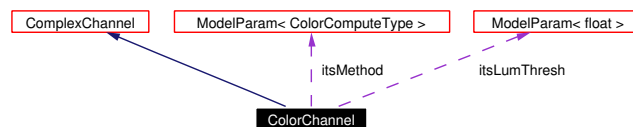
A double opponent color channel class that combines r/g, b/y subchannels.

```
#include <Channels.H>
```

Inheritance diagram for ColorChannel:



Collaboration diagram for ColorChannel:



### Public Member Functions

- **ColorChannel** (**ModelManager** &mgr)  
*Construct with standard params.*
- virtual **~ColorChannel** ()  
*Destructor.*
- virtual **RedGreenChannel** & **rg** () const  
*Get access the **RedGreenChannel** subchannel.*
- virtual **BlueYellowChannel** & **by** () const  
*Get access to the **BlueYellowChannel** subchannel.*

- virtual void [exportOptions](#) (const int whatToExport, const bool recurse=true)

*Export some of our [ModelParam](#) parameters as command-line-options.*

## Protected Member Functions

- virtual void [doInput](#) (const [Image](#)< [PixRGB](#)< float > > \*colimg, const [Image](#)< float > \*, const double t, const [Image](#)< byte > &clipMask)

*ColorChannel requires only color input.*

## Protected Attributes

- [ModelParam](#)< float > [itsLumThresh](#)

*Luminance threshold (see [getRGBY\(\)](#)).*

- [ModelParam](#)< [ColorComputeType](#) > [itsMethod](#)
- [uint](#) [itsRGidx](#)
- [uint](#) [itsBYidx](#)

### 10.52.1 Detailed Description

A double opponent color channel class that combines r/g, b/y subchannels.

This is an example of an heterogeneous complex channel where the various subchannels have distinct visual features. Thus, the [JetSpec](#) representation for such complex channel will be a series of independent single-channel representations, one for each subchannel.

Definition at line 932 of file Channels.H.

### 10.52.2 Constructor & Destructor Documentation

#### 10.52.2.1 [ColorChannel::ColorChannel](#) ([ModelManager](#) & mgr)

Construct with standard params.

Definition at line 820 of file Channels.C.

References [ComplexChannel::addSubChan\(\)](#), [COLOR](#), [ColorComputeType](#), [COL-standard](#), and [makeShared\(\)](#).

**10.52.2.2 ColorChannel::~~ColorChannel () [virtual]**

Destructor.

Definition at line 846 of file Channels.C.

**10.52.3 Member Function Documentation****10.52.3.1 BlueYellowChannel & ColorChannel::by () const [virtual]**

Get access to the [BlueYellowChannel](#) subchannel.

Reimplemented in [ColorChannelSurprise](#).

Definition at line 838 of file Channels.C.

References [ComplexChannel::subChan\(\)](#).

Referenced by [doInput\(\)](#).

**10.52.3.2 void ColorChannel::doInput (const [Image](#)< [PixRGB](#)< float > > \* colimg, const [Image](#)< float > \*, const double t, const [Image](#)< byte > & clipMask) [protected, virtual]**

ColorChannel requires only color input.

Implements [ChannelBase](#).

Definition at line 850 of file Channels.C.

References [ASSERT](#), [by\(\)](#), [COLsimple](#), [COLstandard](#), [getRGBY\(\)](#), [getRGBYsimple\(\)](#), [ModelParam< float >::getVal\(\)](#), [ModelParam< ColorComputeType >::getVal\(\)](#), [ChannelBase::input\(\)](#), [rg\(\)](#), and [ComplexChannel::subChan\(\)](#).

**10.52.3.3 void ColorChannel::exportOptions (const int whatToExport, const bool recurse = true) [virtual]**

Export some of our [ModelParam](#) parameters as command-line-options.

See the base function in [ModelComponent.H](#) for info.

Reimplemented from [ComplexChannel](#).

Definition at line 879 of file Channels.C.

References [ComplexChannel::exportOptions\(\)](#), [OPTEXP\\_CORE](#), and [ModelManager::requestOption\(\)](#).

**10.52.3.4 [RedGreenChannel](#) & [ColorChannel::rg \(\) const](#) [virtual]**

Get access the [RedGreenChannel](#) subchannel.

Reimplemented in [ColorChannelSurprise](#).

Definition at line 830 of file Channels.C.

References [ComplexChannel::subChan\(\)](#).

Referenced by [doInput\(\)](#).

**10.52.4 Member Data Documentation****10.52.4.1 [ModelParam<float>](#) [ColorChannel::itsLumThresh](#) [protected]**

Luminance threshold (see [getRGBY\(\)](#)).

Definition at line 953 of file Channels.H.

The documentation for this class was generated from the following files:

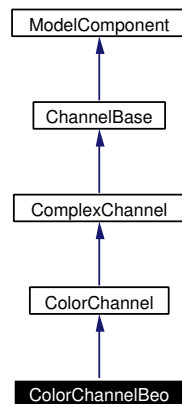
- [Channels.H](#)
- [Channels.C](#)

## 10.53 ColorChannelBeo Class Reference

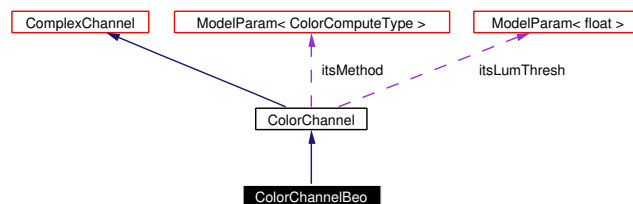
Beowulf version of [ColorChannel](#).

```
#include <ChannelsBeo.H>
```

Inheritance diagram for ColorChannelBeo:



Collaboration diagram for ColorChannelBeo:



### Public Member Functions

- **ColorChannelBeo** ([ModelManager](#) &mgr, const [SharedPtr](#)< [Beowulf](#) > &beow)

#### 10.53.1 Detailed Description

Beowulf version of [ColorChannel](#).

Definition at line 167 of file ChannelsBeo.H.

The documentation for this class was generated from the following files:

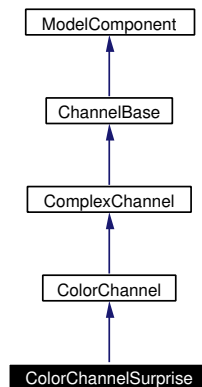
- [ChannelsBeo.H](#)
- [ChannelsBeo.C](#)

## 10.54 ColorChannelSurprise Class Reference

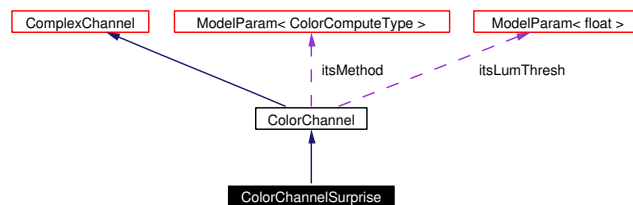
A double opponent color channel class that combines r/g, b/y subchannels.

```
#include <ChannelsSurprise.H>
```

Inheritance diagram for ColorChannelSurprise:



Collaboration diagram for ColorChannelSurprise:



### Public Member Functions

- **ColorChannelSurprise** (**ModelManager** &mgr)  
*Construct with standard params.*
- virtual **~ColorChannelSurprise** ()  
*Destructor.*
- virtual **RedGreenChannelSurprise** & **rg** () const  
*Get access the [RedGreenChannel](#) subchannel.*

- virtual [BlueYellowChannelSurprise](#) & [by](#) () const

*Get access to the [BlueYellowChannel](#) subchannel.*

### 10.54.1 Detailed Description

A double opponent color channel class that combines r/g, b/y subchannels.

Definition at line 452 of file ChannelsSurprise.H.

### 10.54.2 Constructor & Destructor Documentation

#### 10.54.2.1 [ColorChannelSurprise::ColorChannelSurprise](#) ([ModelManager](#) & *mgr*)

Construct with standard params.

Definition at line 445 of file ChannelsSurprise.C.

References [ComplexChannel::addSubChan\(\)](#), [makeShared\(\)](#), [ModelComponent::removeAllSubComponents\(\)](#), and [ChannelBase::setHomogeneous\(\)](#).

#### 10.54.2.2 [ColorChannelSurprise::~~ColorChannelSurprise](#) () [virtual]

Destructor.

Definition at line 458 of file ChannelsSurprise.C.

### 10.54.3 Member Function Documentation

#### 10.54.3.1 [BlueYellowChannelSurprise](#) & [ColorChannelSurprise::by](#) () const [virtual]

Get access to the [BlueYellowChannel](#) subchannel.

Reimplemented from [ColorChannel](#).

Definition at line 466 of file ChannelsSurprise.C.

References [ComplexChannel::subChan\(\)](#).

#### 10.54.3.2 [RedGreenChannelSurprise](#) & [ColorChannelSurprise::rg](#) () const [virtual]

Get access the [RedGreenChannel](#) subchannel.



Reimplemented from [ColorChannel](#).

Definition at line 462 of file ChannelsSurprise.C.

References [ComplexChannel::subChan\(\)](#).

The documentation for this class was generated from the following files:

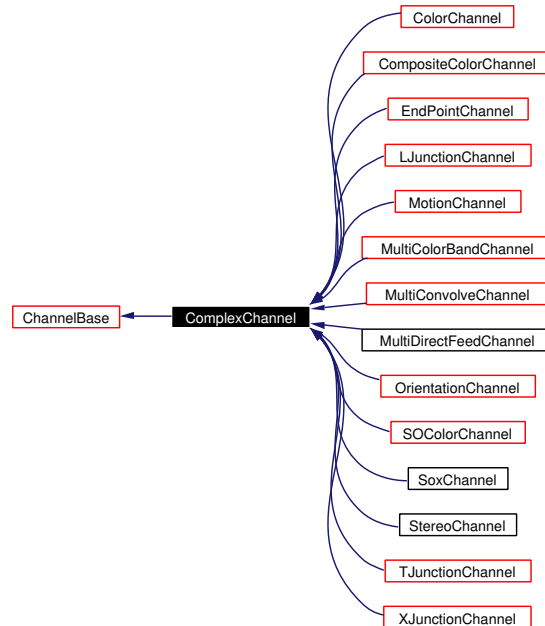
- [ChannelsSurprise.H](#)
- ChannelsSurprise.C

## 10.55 ComplexChannel Class Reference

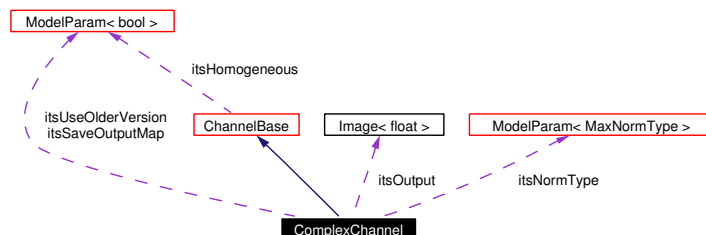
A an abstract channel class that pools across several subchannels.

```
#include <ComplexChannel.H>
```

Inheritance diagram for ComplexChannel:



Collaboration diagram for ComplexChannel:



### Public Member Functions

- **ComplexChannel** (**ModelManager** &mgr, const **std::string** &descrName, const **std::string** &tagName, const **VisualFeature** vs)

Constructor.

- virtual [~ComplexChannel](#) ()  
Virtual destructor.
- virtual void [exportOptions](#) (const int whatToExport, const bool recurse=true)  
Export some of our [ModelParam](#) parameters as command-line-options.
- virtual void [reset](#) (const bool recurse=true)  
Reset ComplexChannel and propagate the reset signal to the sub-components.
- virtual [uint numChans](#) () const  
Get the number of subchannels.
- virtual [ChannelBase & subChan](#) (const [uint](#) idx) const  
Look up one of the subchannels by its integer index.
- virtual [ChannelBase & subChanFromSubmapNum](#) (const [uint](#) oldIdx, [uint](#) &newIdx) const  
Look up one of the subchannels by a submap number.
- virtual [uint addSubChan](#) (const [SharedPtr](#)< [ChannelBase](#) > ch)  
Add a subchannel.
- virtual void [readFrom](#) (const [ParamMap](#) &pmap)  
Read params from the [ParamMap](#).
- virtual void [writeTo](#) ([ParamMap](#) &pmap) const  
Write params to the [ParamMap](#).
- virtual void [setListener](#) ([SharedPtr](#)< [ChannelListener](#) > listener)  
Set a [ChannelListener](#) for this channel.
- virtual void [cleanCoeffs](#) (const double eps)  
Set any coefficients less than eps to zero.
- virtual void [clampCoeffs](#) (const double cmin, const double cmax)  
Clamp the weighting coefficients to the specified range.
- virtual double [absSumCoeffs](#) () const  
Return the sum of the absolute values of the weighting coefficients.
- virtual void [normalizeCoeffs](#) (const double div)

*Normalize the weighting coefficients by dividing by div.*

- virtual void **learn** (const **Image**< **byte** > &dmap, const double eta, const bool softmax=false, const int inthresh=20, const int outthresh=255)

*Adjust the weighting coefficients according to the target map.*

- virtual bool **outputAvailable** () const

*Query whether the channel is ready to give yield valid output.*

- virtual const **Dims** & **getMapDims** () const

*Return the dimensions of the output image.*

- virtual const **Dims** & **getInputDims** () const

*Return the dimensions of the most recent input image.*

- virtual void **fillJet** (const **Point2D** &loc, **Jet**< float > &j, **std::vector**< int > &v) const

*Fill the components of the given Jet that relate to this channel.*

- virtual void **fillJetNI** (const **Point2D** &loc, **Jet**< float > &j, **std::vector**< int > &v) const

*Fill the components of the given Jet that relate to this channel.*

- virtual void **fillJetSpec** (**JetSpec** &js) const

*Fill the specification of a Jet with data that relates to this channel.*

- virtual **uint** **numSubmaps** () const

*Return the number of feature maps represented by this channel.*

- virtual **Image**< float > **getSubmap** (const **uint** index) const

*Get the feature map at the given index.*

- virtual const char \* **getSubmapName** (const **uint** index) const

*Get the name/description of the feature map at the given index.*

- virtual void **biasFeatures** (const double \*mean, const double \*sigma, int &idx)

- virtual void **getFeatures** (const **Point2D** &locn, **std::vector**< double > &mean) const

- virtual void **findWeight** ()

*Weight of the channel is the maximum weight of its subchannels.*

- virtual bool **reco** (const **ParamMap** &pmap, double &mismatch)

*Recognise this location in the channel by checking if it is obj.*

- virtual void [findLikelihood](#) (const [ParamMap](#) &pmap, double &likelihood)  
*what is the probability that the fixation belongs to this object?*
- virtual [Image](#)< float > [getOutput](#) ()  
*Form output by combining output of subchannels.*
- virtual void [drawResults](#) ([Image](#)< [PixRGB](#)< byte > > &traj)  
*Draws something channel-specific in the trajectory image.*
- virtual void [saveResults](#) (const [SharedPtr](#)< [OutputFrameSeries](#) > &ofs)  
*Save our various maps using an [OutputFrameSeries](#).*

## Protected Member Functions

- virtual void [killCaches](#) ()  
*Clears the output cache.*
- virtual [Image](#)< float > [combineOutputs](#) ()  
*Combines the outputs of subchannels.*
- virtual void [start2](#) ()  
*get started*

## Protected Attributes

- [ModelParam](#)< [MaxNormType](#) > [itsNormType](#)  
*Normalization; see [fancynorm.H](#).*
- [ModelParam](#)< bool > [itsSaveOutputMap](#)  
*Save our output map?*
- [ModelParam](#)< bool > [itsUseOlderVersion](#)

### 10.55.1 Detailed Description

A an abstract channel class that pools across several subchannels.

Definition at line 47 of file ComplexChannel.H.

## 10.55.2 Constructor & Destructor Documentation

### 10.55.2.1 `ComplexChannel::ComplexChannel (ModelManager & mgr, const std::string & descrName, const std::string & tagName, const VisualFeature vs)`

Constructor.

**Parameters:**

- mgr* our [ModelManager](#) (see [ModelManager.H](#))
- descrName* descriptive name for human use
- tagName* name for [ParamMap](#) usage
- vs* The [VisualFeature](#) implemented by the channel

Definition at line 58 of file `ComplexChannel.C`.

References [MaxNormType](#), and [VCXNORM\\_FANCY](#).

### 10.55.2.2 `ComplexChannel::~~ComplexChannel ()` [virtual]

Virtual destructor.

Definition at line 68 of file `ComplexChannel.C`.

## 10.55.3 Member Function Documentation

### 10.55.3.1 `double ComplexChannel::absSumCoeffs () const` [virtual]

Return the sum of the absolute values of the weighting coefficients.

Implements [ChannelBase](#).

Definition at line 184 of file `ComplexChannel.C`.

References [ChannelBase::absSumCoeffs\(\)](#), [ChannelBase::getTotalWeight\(\)](#), [numChans\(\)](#), [subChan\(\)](#), and [uint](#).

### 10.55.3.2 `uint ComplexChannel::addSubChan (const SharedPtr< ChannelBase > ch)` [virtual]

Add a subchannel.

This is just a wrapper to [ModelComponent::addSubComponent\(\)](#), with some checking that the subcomponent indeed is a [ChannelBase](#) derivative, and some adjustment of our homogeneity (if the newly added subchannel has a different [VisualFeature](#) from ours, then be become heterogeneous). Returns the `subChan` index.

Definition at line 411 of file ComplexChannel.C.

References [ModelComponent::addSubComponent\(\)](#), [ChannelBase::setHomogeneous\(\)](#), [uint](#), and [ChannelBase::visualFeature\(\)](#).

Referenced by [SoxChannel::buildSubChans\(\)](#), [EndPointChannelSurprise::buildSubChans\(\)](#), [XJunctionChannelSurprise::buildSubChans\(\)](#), [TJunctionChannelSurprise::buildSubChans\(\)](#), [LJunctionChannelSurprise::buildSubChans\(\)](#), [MotionChannelSurprise::buildSubChans\(\)](#), [MultiColorBandChannelSurprise::buildSubChans\(\)](#), [OrientationChannelSurprise::buildSubChans\(\)](#), [MotionChannelBeo::buildSubChans\(\)](#), [MultiColorBandChannelBeo::buildSubChans\(\)](#), [OrientationChannelBeo::buildSubChans\(\)](#), [StereoChannel::buildSubChans\(\)](#), [EndPointChannel::buildSubChans\(\)](#), [XJunctionChannel::buildSubChans\(\)](#), [TJunctionChannel::buildSubChans\(\)](#), [LJunctionChannel::buildSubChans\(\)](#), [MotionChannel::buildSubChans\(\)](#), [MultiColorBandChannel::buildSubChans\(\)](#), [OrientationChannel::buildSubChans\(\)](#), [ColorChannel::ColorChannel\(\)](#), [ColorChannelSurprise::ColorChannelSurprise\(\)](#), [CompositeColorChannel::CompositeColorChannel\(\)](#), [CompositeColorChannelSurprise::CompositeColorChannelSurprise\(\)](#), [MultiConvolveChannel::MultiConvolveChannel\(\)](#), [MultiConvolveChannelSurprise::MultiConvolveChannelSurprise\(\)](#), [MultiDirectFeedChannel::setNumChans\(\)](#), [SOColorChannel::SOColorChannel\(\)](#), and [SOColorChannelSurprise::SOColorChannelSurprise\(\)](#).

### 10.55.3.3 void ComplexChannel::biasFeatures (const double \* mean, const double \* sigma, int & idx) [virtual]

Bias the features at submap idx with mean, intra-class std. dev and inter-sub-class std. dev

Implements [ChannelBase](#).

Definition at line 336 of file ComplexChannel.C.

References [ChannelBase::biasFeatures\(\)](#), [numChans\(\)](#), [subChan\(\)](#), and [uint](#).

### 10.55.3.4 void ComplexChannel::clampCoeffs (const double cmin, const double cmax) [virtual]

Clamp the weighting coefficients to the specified range.

Implements [ChannelBase](#).

Definition at line 177 of file ComplexChannel.C.

References [ChannelBase::clampCoeffs\(\)](#), [numChans\(\)](#), [subChan\(\)](#), and [uint](#).

### 10.55.3.5 void ComplexChannel::cleanCoeffs (const double eps) [virtual]

Set any coefficients less than eps to zero.

Implements [ChannelBase](#).

Definition at line 170 of file ComplexChannel.C.

References [ChannelBase::cleanCoeffs\(\)](#), [numChans\(\)](#), [subChan\(\)](#), and [uint](#).

#### 10.55.3.6 [Image](#)< float > [ComplexChannel::combineOutputs \(\)](#) [protected, virtual]

Combines the outputs of subchannels.

The default implementation just sums the outputs of the subchannels and then [max-Normalize\(\)](#)'s if requested by itsNormType.

Reimplemented in [MultiDirectFeedChannel](#), and [SoxChannel](#).

Definition at line 435 of file ComplexChannel.C.

References [ModelManager::debugMode\(\)](#), [ModelComponent::descriptiveName\(\)](#), [getMapDims\(\)](#), [Image< T >::getMinMax\(\)](#), [ChannelBase::getOutput\(\)](#), [ChannelBase::getTotalWeight\(\)](#), [ModelParam< float >::getVal\(\)](#), [ModelParam< MaxNormType >::getVal\(\)](#), [goodness\\_map\(\)](#), [itsNormType](#), [maxNormalize\(\)](#), [maxNormTypeName\(\)](#), [numChans\(\)](#), [subChan\(\)](#), [ModelComponent::tagName\(\)](#), [uint](#), [VCXNORM\\_-LANDMARK](#), and [ZEROS](#).

Referenced by [getOutput\(\)](#).

#### 10.55.3.7 [void ComplexChannel::drawResults \(Image< PixRGB< byte > > & traj\)](#) [virtual]

Draws something channel-specific in the trajectory image.

By default, we do nothing but propagate the call to our subchannels

Reimplemented from [ChannelBase](#).

Definition at line 482 of file ComplexChannel.C.

References [ChannelBase::drawResults\(\)](#), [numChans\(\)](#), [subChan\(\)](#), and [uint](#).

#### 10.55.3.8 [void ComplexChannel::exportOptions \(const int whatToExport, const bool recurse = true\)](#) [virtual]

Export some of our [ModelParam](#) parameters as command-line-options.

See the base function in [ModelComponent.H](#) for info.

Reimplemented from [ChannelBase](#).

Reimplemented in [ColorChannel](#), [OrientationChannel](#), [MultiColorBandChannel](#),



[MotionChannel](#), [StereoChannel](#), [LJunctionChannel](#), [TJunctionChannel](#), [XJunctionChannel](#), [EndPointChannel](#), and [SoxChannel](#).

Definition at line 89 of file ComplexChannel.C.

References [ChannelBase::exportOptions\(\)](#), [itsNormType](#), [itsSaveOutputMap](#), [OPTEXP\\_CORE](#), [OPTEXP\\_SAVE](#), and [ModelManager::requestOption\(\)](#).

Referenced by [SoxChannel::exportOptions\(\)](#), [StereoChannel::exportOptions\(\)](#), [EndPointChannel::exportOptions\(\)](#), [XJunctionChannel::exportOptions\(\)](#), [TJunctionChannel::exportOptions\(\)](#), [LJunctionChannel::exportOptions\(\)](#), [MotionChannel::exportOptions\(\)](#), [MultiColorBandChannel::exportOptions\(\)](#), [OrientationChannel::exportOptions\(\)](#), and [ColorChannel::exportOptions\(\)](#).

### 10.55.3.9 void ComplexChannel::fillJet (const [Point2D](#) & loc, [Jet](#)< float > & j, [std::vector](#)< int > & v) const [virtual]

Fill the components of the given [Jet](#) that relate to this channel.

All relevant components of the [Jet](#) are set to this channel's values at the specified location across all pyramid levels. The channel will store its data starting at the given index in the jet, and will return the number of entries filled. The index vector will be modified as the various [Jet](#) components are explored, but will be restored to its original state upon completion.

Implements [ChannelBase](#).

Definition at line 227 of file ComplexChannel.C.

References [ChannelBase::fillJet\(\)](#), [ChannelBase::isHomogeneous\(\)](#), [numChans\(\)](#), [subChan\(\)](#), and [uint](#).

### 10.55.3.10 void ComplexChannel::fillJetNI (const [Point2D](#) & loc, [Jet](#)< float > & j, [std::vector](#)< int > & v) const [virtual]

Fill the components of the given [Jet](#) that relate to this channel.

This version does not attempt to use interpolation

Implements [ChannelBase](#).

Definition at line 247 of file ComplexChannel.C.

References [ChannelBase::fillJetNI\(\)](#), [ChannelBase::isHomogeneous\(\)](#), [numChans\(\)](#), [subChan\(\)](#), and [uint](#).

#### 10.55.3.11 **void ComplexChannel::fillJetSpec ([JetSpec](#) & js) const** [virtual]

Fill the specification of a [Jet](#) with data that relates to this channel.

Implements [ChannelBase](#).

Definition at line 267 of file ComplexChannel.C.

References [JetSpec::addIndexRange\(\)](#), [ChannelBase::fillJetSpec\(\)](#), [ChannelBase::isHomogeneous\(\)](#), [numChans\(\)](#), [RAW](#), [subChan\(\)](#), [uint](#), and [ChannelBase::visualFeature\(\)](#).

#### 10.55.3.12 **void ComplexChannel::findLikelihood (const [ParamMap](#) & pmap, double & likelihood) [virtual]**

what is the probability that the fixation belongs to this object?

Implements [ChannelBase](#).

Definition at line 393 of file ComplexChannel.C.

References [ChannelBase::findLikelihood\(\)](#), [ParamMap::getSubmap\(\)](#), [ParamMap::hasParam\(\)](#), [numChans\(\)](#), [subChan\(\)](#), [ModelComponent::tagName\(\)](#), and [uint](#).

#### 10.55.3.13 **void ComplexChannel::findWeight () [virtual]**

Weight of the channel is the maximum weight of its subchannels.

used for biasing to find targets

Implements [ChannelBase](#).

Definition at line 352 of file ComplexChannel.C.

References [ChannelBase::findWeight\(\)](#), [ChannelBase::getTotalWeight\(\)](#), [numChans\(\)](#), [ChannelBase::setTotalWeight\(\)](#), [subChan\(\)](#), and [uint](#).

#### 10.55.3.14 **const [Dims](#) & ComplexChannel::getInputDims () const** [virtual]

Return the dimensions of the most recent input image.

Implements [ChannelBase](#).

Definition at line 223 of file ComplexChannel.C.

References [ChannelBase::getInputDims\(\)](#), and [subChan\(\)](#).

Referenced by [SoxChannel::getNonlinearResponse\(\)](#).

**10.55.3.15** `const Dims & ComplexChannel::getMapDims () const` [virtual]

Return the dimensions of the output image.

Implements [ChannelBase](#).

Definition at line 219 of file ComplexChannel.C.

References [ChannelBase::getMapDims\(\)](#), and [subChan\(\)](#).

Referenced by [SoxChannel::combineOutputs\(\)](#), [MultiDirectFeedChannel::combineOutputs\(\)](#), and [combineOutputs\(\)](#).

**10.55.3.16** `Image< float > ComplexChannel::getOutput ()` [virtual]

Form output by combining output of subchannels.

This function just provides a caching wrapper around [combineOutputs\(\)](#), where the real work is done. [ComplexChannel](#) provides a default implementation of [combineOutputs\(\)](#) that sums the outputs of the subchannels and then optionally [max-Normalize\(\)](#)'s the result.

Implements [ChannelBase](#).

Reimplemented in [OrientationChannel](#).

Definition at line 428 of file ComplexChannel.C.

References [combineOutputs\(\)](#), and [Image< float >::initialized\(\)](#).

Referenced by [OrientationChannel::getOutput\(\)](#), and [saveResults\(\)](#).

**10.55.3.17** `Image< float > ComplexChannel::getSubmap (const uint index)  
const` [virtual]

Get the feature map at the given index.

This function is intended to be implemented by subclasses so that it recurses into any nested channels, mapping nested submaps into a linear sequence of indices.

Implements [ChannelBase](#).

Definition at line 320 of file ComplexChannel.C.

References [ChannelBase::getSubmap\(\)](#), [subChan\(\)](#), and [uint](#).

**10.55.3.18** `const char * ComplexChannel::getSubmapName (const uint index)  
const` [virtual]

Get the name/description of the feature map at the given index.

This function is intended to be implemented by subclasses so that it recurses into any nested channels, mapping nested submaps into a linear sequence of indices. The output of this function is intended only to be human-readable (e.g. for labeling image displays), and does not have to follow any particular parseable syntax.

Implements [ChannelBase](#).

Definition at line 328 of file `ComplexChannel.C`.

References `ChannelBase::getSubmapName()`, `subChan()`, and `uint`.

#### 10.55.3.19 `void ComplexChannel::killCaches ()` [protected, virtual]

Clears the output cache.

use old version? see `Visualcortex.H`

Reimplemented from [ChannelBase](#).

Reimplemented in [SoxChannel](#).

Definition at line 421 of file `ComplexChannel.C`.

References `Image< float >::freeMem()`, and `ChannelBase::killCaches()`.

Referenced by `SoxChannel::killCaches()`, and `learn()`.

#### 10.55.3.20 `void ComplexChannel::learn (const Image< byte > & dmap, const double eta, const bool softmax = false, const int inthresh = 20, const int outhresh = 255)` [virtual]

Adjust the weighting coefficients according to the target map.

Implements [ChannelBase](#).

Definition at line 200 of file `ComplexChannel.C`.

References `killCaches()`, `ChannelBase::learn()`, `numChans()`, `subChan()`, and `uint`.

#### 10.55.3.21 `void ComplexChannel::normalizeCoeffs (const double div)` [virtual]

Normalize the weighting coefficients by dividing by `div`.

Implements [ChannelBase](#).

Definition at line 194 of file `ComplexChannel.C`.

References `ChannelBase::normalizeCoeffs()`, `numChans()`, `subChan()`, and `uint`.

**10.55.3.22** [uint](#) **ComplexChannel::numChans () const** [virtual]

Get the number of subchannels.

Implements [ChannelBase](#).

Definition at line 115 of file ComplexChannel.C.

References [ModelComponent::numSubComp\(\)](#), and [uint](#).

Referenced by [absSumCoeffs\(\)](#), [biasFeatures\(\)](#), [clampCoeffs\(\)](#), [cleanCoeffs\(\)](#), [SoxChannel::combineOutputs\(\)](#), [MultiDirectFeedChannel::combineOutputs\(\)](#), [combineOutputs\(\)](#), [MultiDirectFeedChannel::directFeed\(\)](#), [SoxChannel::doInput\(\)](#), [MultiDirectFeedChannel::doInput\(\)](#), [EndPointChannel::doInput\(\)](#), [XJunctionChannel::doInput\(\)](#), [TJunctionChannel::doInput\(\)](#), [LJunctionChannel::doInput\(\)](#), [MotionChannel::doInput\(\)](#), [MultiColorBandChannel::doInput\(\)](#), [OrientationChannel::doInput\(\)](#), [CompositeColorChannel::doInput\(\)](#), [MultiConvolveChannel::doInput\(\)](#), [OrientationChannel::doInteractions\(\)](#), [drawResults\(\)](#), [fillJet\(\)](#), [fillJetNI\(\)](#), [fillJetSpec\(\)](#), [findLikelihood\(\)](#), [findWeight\(\)](#), [SoxChannel::getNonlinearResponse\(\)](#), [MultiDirectFeedChannel::inputPyramid\(\)](#), [MultiDirectFeedChannel::inputPyramidVector\(\)](#), [learn\(\)](#), [normalizeCoeffs\(\)](#), [numSubmaps\(\)](#), [outputAvailable\(\)](#), [readFrom\(\)](#), [reco\(\)](#), [saveResults\(\)](#), [setListener\(\)](#), [SoxChannel::setModelParamString\(\)](#), [StereoChannel::setModelParamString\(\)](#), [EndPointChannel::setModelParamString\(\)](#), [XJunctionChannel::setModelParamString\(\)](#), [TJunctionChannel::setModelParamString\(\)](#), [LJunctionChannel::setModelParamString\(\)](#), [MotionChannel::setModelParamString\(\)](#), [MultiColorBandChannel::setModelParamString\(\)](#), [OrientationChannel::setModelParamString\(\)](#), [MultiDirectFeedChannel::setNumChans\(\)](#), [start2\(\)](#), and [writeTo\(\)](#).

**10.55.3.23** [uint](#) **ComplexChannel::numSubmaps () const** [virtual]

Return the number of feature maps represented by this channel.

Implements [ChannelBase](#).

Definition at line 291 of file ComplexChannel.C.

References [numChans\(\)](#), [ChannelBase::numSubmaps\(\)](#), [subChan\(\)](#), and [uint](#).

Referenced by [subChanFromSubmapNum\(\)](#).

**10.55.3.24** **bool** **ComplexChannel::outputAvailable () const** [virtual]

Query whether the channel is ready to give yield valid output.

In some cases (e.g. with motion channels), it may take several input frames before the channel is fully initialized.

Implements [ChannelBase](#).

Definition at line 211 of file ComplexChannel.C.

References numChans(), ChannelBase::outputAvailable(), subChan(), and uint.

#### 10.55.3.25 void ComplexChannel::readFrom (const ParamMap & pmap) [virtual]

Read params from the ParamMap.

Reimplemented from ChannelBase.

Definition at line 134 of file ComplexChannel.C.

References ParamMap::getSubmap(), ParamMap::hasParam(), numChans(), ChannelBase::readFrom(), subChan(), ModelComponent::tagName(), and uint.

#### 10.55.3.26 bool ComplexChannel::reco (const ParamMap & pmap, double & mismatch) [virtual]

Recognise this location in the channel by checking if it is obj.

Implements ChannelBase.

Definition at line 376 of file ComplexChannel.C.

References ParamMap::getSubmap(), ParamMap::hasParam(), numChans(), ChannelBase::reco(), subChan(), ModelComponent::tagName(), and uint.

#### 10.55.3.27 void ComplexChannel::reset (const bool recurse = true) [virtual]

Reset ComplexChannel and propagate the reset signal to the sub-components.

See the base function in ModelComponent.H for info.

Reimplemented from ModelComponent.

Definition at line 105 of file ComplexChannel.C.

References Image< float >::freeMem(), and ModelComponent::reset().

#### 10.55.3.28 void ComplexChannel::saveResults (const SharedPtr< OutputFrameSeries > & ofs) [virtual]

Save our various maps using an OutputFrameSeries.

By default, we save our output map (if itsSaveOutputMap is true) and propagate the call to our subchannels. Our output (conspicuity) map will be prefixed by "CO<tagname>-"

Reimplemented from [ChannelBase](#).

Definition at line 486 of file ComplexChannel.C.

References [getOutput\(\)](#), [ModelParam< bool >::getVal\(\)](#), [itsSaveOutputMap](#), [numChans\(\)](#), [ChannelBase::saveResults\(\)](#), [subChan\(\)](#), [ModelComponent::tagName\(\)](#), and [uint](#).

### 10.55.3.29 void ComplexChannel::setListener ([SharedPtr< ChannelListener > listener](#)) [virtual]

Set a [ChannelListener](#) for this channel.

The [ChannelListener](#) which will get called once for each center/surround pairing.

Implements [ChannelBase](#).

Definition at line 163 of file ComplexChannel.C.

References [numChans\(\)](#), [ChannelBase::setListener\(\)](#), [subChan\(\)](#), and [uint](#).

### 10.55.3.30 void ComplexChannel::start2 () [protected, virtual]

get started

Reimplemented from [ModelComponent](#).

Definition at line 72 of file ComplexChannel.C.

References [ModelParam< MaxNormType >::getVal\(\)](#), [ModelParam< bool >::getVal\(\)](#), [itsNormType](#), [numChans\(\)](#), [ChannelBase::setOutputRange\(\)](#), [subChan\(\)](#), [uint](#), and [VCXNORM\\_MAXNORM](#).

### 10.55.3.31 [ChannelBase](#) & ComplexChannel::subChan (const [uint idx](#)) const [virtual]

Look up one of the subchannels by its integer index.

Reimplemented in [SoxChannel](#).

Definition at line 119 of file ComplexChannel.C.

References [SharedPtr< ModelComponent >::get\(\)](#), [ModelComponent::subComponent\(\)](#), and [uint](#).

Referenced by [absSumCoeffs\(\)](#), [SOColorChannelSurprise::b\(\)](#), [SOColorChannel::b\(\)](#), [MultiColorBandChannelSurprise::band\(\)](#), [MultiColorBandChannel::band\(\)](#), [biasFeatures\(\)](#), [SOColorChannelSurprise::by\(\)](#), [ColorChannelSurprise::by\(\)](#), [SOColorChannel::by\(\)](#), [ColorChannel::by\(\)](#), [clampCoeffs\(\)](#), [cleanCoeffs\(\)](#), [combineOutputs\(\)](#), [MotionChannelSurprise::dirChan\(\)](#), [MotionChannel::dirChan\(\)](#), [MultiDirectFeed-](#)

Channel::directFeed(), StereoChannel::dispChan(), EndPointChannel::doInput(), XJunctionChannel::doInput(), TJunctionChannel::doInput(), LJunctionChannel::doInput(), JunctionChannel::doInput(), MotionChannel::doInput(), MultiColorBandChannel::doInput(), SOColorChannel::doInput(), ColorChannel::doInput(), MultiConvolveChannel::doInput(), drawResults(), fillJet(), fillJetNI(), fillJetSpec(), findLikelihood(), findWeight(), SOColorChannelSurprise::g(), SOColorChannel::g(), OrientationChannelSurprise::gabor(), OrientationChannel::gabor(), getInputDims(), getMapDims(), getSubmap(), getSubmapName(), SOColorChannelSurprise::gr(), SOColorChannel::gr(), learn(), normalizeCoeffs(), numSubmaps(), outputAvailable(), SOColorChannelSurprise::r(), SOColorChannel::r(), readFrom(), reco(), SOColorChannelSurprise::rg(), ColorChannelSurprise::rg(), SOColorChannel::rg(), ColorChannel::rg(), saveResults(), setListener(), MultiConvolveChannel::start1(), start2(), SoxChannel::subChan(), subChanFromSubmapNum(), writeTo(), SOColorChannelSurprise::yb(), SOColorChannel::yb(), and SOColorChannel::yb().

#### 10.55.3.32 [ChannelBase](#) & `ComplexChannel::subChanFromSubmapNum` (const [uint](#) *oldIdx*, [uint](#) & *newIdx*) const [virtual]

Look up one of the subchannels by a submap number.

##### Parameters:

*oldIdx* - the submap index (in the `ComplexChannel`);

*newIdx* - return value: the number of the submap within the subchannel

Definition at line 123 of file `ComplexChannel.C`.

References `ASSERT`, `numSubmaps()`, `subChan()`, and `uint`.

Referenced by `ShapeEstimator::getSingleChannel()`.

#### 10.55.3.33 `void ComplexChannel::writeTo` ([ParamMap](#) & *pmap*) const [virtual]

Write params to the [ParamMap](#).

Reimplemented from [ChannelBase](#).

Definition at line 150 of file `ComplexChannel.C`.

References `numChans()`, `ParamMap::putSubmap()`, `subChan()`, `ModelComponent::tagName()`, `uint`, and `ChannelBase::writeTo()`.



## 10.55.4 Member Data Documentation

### 10.55.4.1 [ModelParam<MaxNormType> ComplexChannel::itsNormType](#) [protected]

Normalization; see [fancynorm.H](#).

Definition at line 159 of file ComplexChannel.H.

Referenced by `combineOutputs()`, `exportOptions()`, and `start2()`.

### 10.55.4.2 [ModelParam<bool> ComplexChannel::itsSaveOutputMap](#) [protected]

Save our output map?

Definition at line 160 of file ComplexChannel.H.

Referenced by `exportOptions()`, and `saveResults()`.

The documentation for this class was generated from the following files:

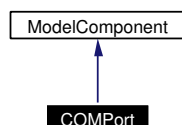
- [ComplexChannel.H](#)
- [ComplexChannel.C](#)

## 10.56 COMPort Class Reference

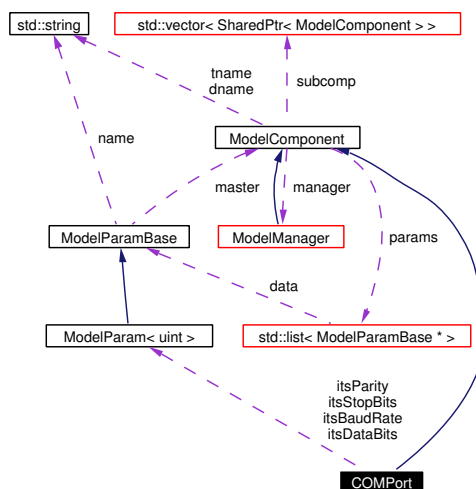
A simple class for interfacing with a serial (COM) port.

```
#include <COMPort.H>
```

Inheritance diagram for COMPort:



Collaboration diagram for COMPort:



### Public Member Functions

- **COMPort** (**ModelManager** &mgr, const **std::string** &descrName="COM Port", const **std::string** &tagName="COMPort")  
*constructor*
- **~COMPort** ()  
*Destructor (closes port if it isn't closed yet).*
- bool **open** (unsigned int access=O\_RDONLY)

*open the com port*

- bool [close](#) (void)  
*close the port*
- bool [write](#) (const char \*data, int num\_bytes)  
*write data to the port*
- bool [read](#) (char \*buffer, int num\_bytes)  
*read data from the port*
- char [read](#) (void)  
*read one byte from the port*

## Protected Member Functions

- virtual void [start1](#) ()  
*get us started:*

## Protected Attributes

- [ModelParam](#)< [std::string](#) > [itsDevName](#)  
*device name*
- [ModelParam](#)< [uint](#) > [itsBaudRate](#)  
*baud rate*
- [ModelParam](#)< [uint](#) > [itsDataBits](#)  
*number of data bits*
- [ModelParam](#)< [uint](#) > [itsStopBits](#)  
*Number of stop bits.*
- [ModelParam](#)< [uint](#) > [itsParity](#)  
*parity*

### 10.56.1 Detailed Description

A simple class for interfacing with a serial (COM) port.

Definition at line 55 of file COMPort.H.

### 10.56.2 Constructor & Destructor Documentation

#### 10.56.2.1 COMPort::COMPort ([ModelManager](#) & *mgr*, const [std::string](#) & *descrName* = "COM Port", const [std::string](#) & *tagName* = "COMPort")

constructor

Definition at line 42 of file COMPort.C.

#### 10.56.2.2 COMPort::~COMPort ()

Destructor (closes port if it isn't closed yet).

Definition at line 74 of file COMPort.C.

References [close\(\)](#).

### 10.56.3 Member Function Documentation

#### 10.56.3.1 bool COMPort::close (void)

close the port

**Returns:**

- false if some error occurred, true otherwise

Definition at line 150 of file COMPort.C.

Referenced by [~COMPort\(\)](#).

#### 10.56.3.2 bool COMPort::open (unsigned int *access* = O\_RDONLY)

open the com port

**Parameters:**

*access* possible values: O\_RDONLY, O\_WRONLY, O\_RDWR

Definition at line 80 of file COMPort.C.

References `ModelParam< uint >::getVal()`, `itsBaudRate`, `itsDataBits`, `itsParity`, and `itsStopBits`.

#### 10.56.3.3 char COMPort::read (void)

read one byte from the port

Definition at line 234 of file COMPort.C.

#### 10.56.3.4 bool COMPort::read (char \* *buffer*, int *num\_bytes*)

read data from the port

**Parameters:**

*buffer* data buffer to write into

*num\_bytes* number of bytes to read

**Returns:**

- false if some error occurred, true otherwise

Definition at line 198 of file COMPort.C.

#### 10.56.3.5 void COMPort::start1 () [protected, virtual]

get us started:

Reimplemented from [ModelComponent](#).

Definition at line 54 of file COMPort.C.

References `ModelParam< uint >::getVal()`, `itsBaudRate`, `itsStopBits`, and `ModelParam< uint >::setVal()`.

#### 10.56.3.6 bool COMPort::write (const char \* *data*, int *num\_bytes*)

write data to the port

**Parameters:**

*data* data buffer to write to the port

*num\_bytes* number of bytes to write

**Returns:**

- false if some error occurred, true otherwise

Definition at line 173 of file COMPort.C.

## 10.56.4 Member Data Documentation

### 10.56.4.1 **ModelParam<uint> COMPort::itsBaudRate** [protected]

baud rate

Possible values: B0, B50, B75, B110, B134, B150, B200, B300, B600, B1200, B2400, B4800, B9600, B19200, B38400 ... and possibly more, depending on your system - see your /usr/include/asm/termbits.h file.

Definition at line 109 of file COMPort.H.

Referenced by open(), and start1().

### 10.56.4.2 **ModelParam<uint> COMPort::itsDataBits** [protected]

number of data bits

possible values: CS5, CS6, CS7, CS8

Definition at line 113 of file COMPort.H.

Referenced by open().

### 10.56.4.3 **ModelParam<std::string> COMPort::itsDevName** [protected]

device name

The COM device to use - Make sure that you have read and write access to the device!

Here are the device paths for the most common UNIX OS:

| System        | COM 1         | COM 2         |
|---------------|---------------|---------------|
| Linux         | "/dev/ttyS0"  | "/dev/ttyS1"  |
| Solaris/SunOS | "/dev/ttya"   | "/dev/ttyb"   |
| IRIX          | "/dev/ttyf1"  | "/dev/ttyf2"  |
| HP-UX         | "/dev/tty1p0" | "/dev/tty2p0" |
| Digital UNIX  | "/dev/tty01"  | "/dev/tty02"  |

Definition at line 102 of file COMPort.H.

### 10.56.4.4 **ModelParam<uint> COMPort::itsParity** [protected]

parity

possible values: NO\_PARITY, EVEN\_PARITY, ODD\_PARITY

Definition at line 121 of file COMPort.H.

Referenced by open().

#### 10.56.4.5 [ModelParam<uint>](#) [COMPort::itsStopBits](#) [protected]

Number of stop bits.

Possible values: STOP\_BITS1, STOP\_BITS2

Definition at line 117 of file COMPort.H.

Referenced by open(), and start1().

The documentation for this class was generated from the following files:

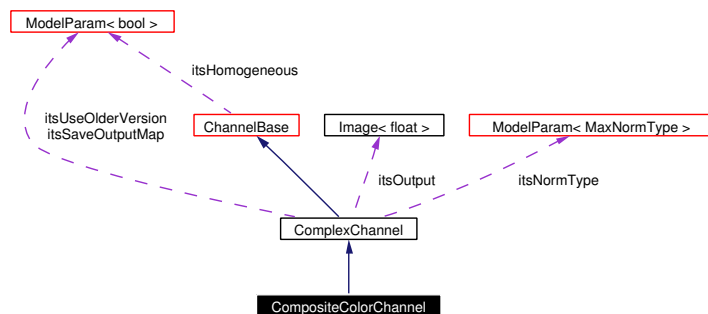
- [COMPort.H](#)
- [COMPort.C](#)

## 10.57 CompositeColorChannel Class Reference

A composite color channel class that contains single and double !

```
#include <Channels.H>
```

Inheritance diagram for CompositeColorChannel: Collaboration diagram for CompositeColorChannel:



### Public Member Functions

- [CompositeColorChannel](#) ([ModelManager](#) &mgr, const double dw=1.0, const double sw=1.0)
- virtual [~CompositeColorChannel](#) ()

*Destructor.*

- [ChannelBase](#) & [channel](#) (const uint idx) const  
*lookup the member channel by its index*
- [ChannelBase](#) & [channel](#) (const char \*name) const  
*lookup the member channel by its name*

### Protected Member Functions

- void [doInput](#) (const [Image](#)< [PixRGB](#)< float > > \*inp, const [Image](#)< float > \*bwimg, const double t, const [Image](#)< byte > &clipMask)

*Computes the double and single opponent color channels.*



## Protected Attributes

- double [itsDoubleOpponent](#)  
*member color channels*
- double [itsSingleOpponent](#)

### 10.57.1 Detailed Description

A composite color channel class that contains single and double !

Definition at line 896 of file Channels.H.

### 10.57.2 Constructor & Destructor Documentation

#### 10.57.2.1 CompositeColorChannel::CompositeColorChannel ([ModelManager](#) & *mgr*, const double *dw* = 1.0, const double *sw* = 1.0)

Construct with standard params. and weight of the double opponent and single opponent channels

Definition at line 775 of file Channels.C.

References [ComplexChannel::addSubChan\(\)](#), [channel\(\)](#), [COMPOSITECOLOR](#), [makeShared\(\)](#), [ChannelBase::setTotalWeight\(\)](#), and [uint](#).

#### 10.57.2.2 CompositeColorChannel::~~CompositeColorChannel () [virtual]

Destructor.

Definition at line 794 of file Channels.C.

### 10.57.3 Member Function Documentation

#### 10.57.3.1 [ChannelBase](#) & CompositeColorChannel::channel (const char \* *name*) const

lookup the member channel by its name

Definition at line 801 of file Channels.C.

References [ModelComponent::subComponent\(\)](#).

### 10.57.3.2 [ChannelBase](#) & `CompositeColorChannel::channel (const uint idx) const`

lookup the member channel by its index

Definition at line 797 of file Channels.C.

References `ModelComponent::subComponent()`, and `uint`.

Referenced by `CompositeColorChannel()`, `CompositeColorChannelSurprise::CompositeColorChannelSurprise()`, and `doInput()`.

### 10.57.3.3 `void CompositeColorChannel::doInput (const Image< PixRGB< float > > * inp, const Image< float > * bwing, const double t, const Image< byte > & clipMask) [protected, virtual]`

Computes the double and single opponent color channels.

Implements [ChannelBase](#).

Definition at line 805 of file Channels.C.

References `channel()`, `ChannelBase::input()`, `ComplexChannel::numChans()`, and `uint`.

## 10.57.4 Member Data Documentation

### 10.57.4.1 `double CompositeColorChannel::itsDoubleOpponent [protected]`

member color channels

Definition at line 921 of file Channels.H.

The documentation for this class was generated from the following files:

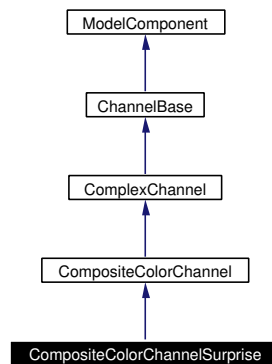
- [Channels.H](#)
- [Channels.C](#)

## 10.58 CompositeColorChannelSurprise Class Reference

A composite color channel class that contains single and double opp.

```
#include <ChannelsSurprise.H>
```

Inheritance diagram for CompositeColorChannelSurprise:



Collaboration diagram for CompositeColorChannelSurprise:

### Public Member Functions

- [CompositeColorChannelSurprise](#) ([ModelManager](#) &mgr, const double dw=1.0, const double sw=1.0)

*Construct with standard params.*

- virtual [~CompositeColorChannelSurprise](#) ()

*Destructor.*

### 10.58.1 Detailed Description

A composite color channel class that contains single and double opp.

Definition at line 438 of file ChannelsSurprise.H.

## 10.58.2 Constructor & Destructor Documentation

### 10.58.2.1 CompositeColorChannelSurprise::CompositeColorChannelSurprise ([ModelManager](#) & *mgr*, const double *dw* = 1.0, const double *sw* = 1.0)

Construct with standard params.

Definition at line 421 of file ChannelsSurprise.C.

References [ComplexChannel::addSubChan\(\)](#), [CompositeColorChannel::channel\(\)](#), [makeShared\(\)](#), [ChannelBase::setTotalWeight\(\)](#), and [uint](#).

### 10.58.2.2 CompositeColorChannelSurprise::~~CompositeColorChannelSurprise () [virtual]

Destructor.

Definition at line 439 of file ChannelsSurprise.C.

The documentation for this class was generated from the following files:

- [ChannelsSurprise.H](#)
- ChannelsSurprise.C

## 10.59 ContourConnection Class Reference

The connection kernel between orientation-tuned units.

### Public Member Functions

- [ContourConnection](#) ([SharedPtr](#)< [ParamMap](#) > pmap)  
*Create the kernel using pmap file parameters.*
- float **weight** (int a, int b, int x, int y) const
- const float \* **weightAXY** (int a, int x, int y) const
- const float \* **excitAXY** (int a, int x, int y) const
- const float \* **inhibAXY** (int a, int x, int y) const
- int **numAngles** () const
- int **sizeX** () const
- int **sizeY** () const
- int **centerX** () const
- int **centerY** () const
- float **angle** (int i) const
- const float \* **angles** () const
- [Image](#)< [byte](#) > **getSummaryImage** () const

### 10.59.1 Detailed Description

The connection kernel between orientation-tuned units.

Definition at line 178 of file ContourChannel.C.

### 10.59.2 Constructor & Destructor Documentation

#### 10.59.2.1 ContourConnection::ContourConnection ([SharedPtr](#)< [ParamMap](#) > pmap)

Create the kernel using pmap file parameters.

Definition at line 259 of file ContourChannel.C.

References [ASSERT](#), [sizeX](#), and [sizeY](#).

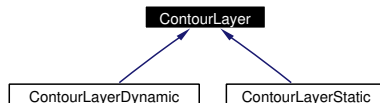
The documentation for this class was generated from the following file:

- ContourChannel.C

## 10.60 ContourLayer Class Reference

Represents a single scale band in which contour facilitation is run.

Inheritance diagram for ContourLayer:



### Public Member Functions

- virtual [Image](#)< float > **iterate** (const [ContourConnection](#) &connection, const char \*savePrefix, const char \*saveSuffix, [Raster::NormalizeMode](#) normMode, bool doSave)=0

#### 10.60.1 Detailed Description

Represents a single scale band in which contour facilitation is run.

Definition at line 597 of file ContourChannel.C.

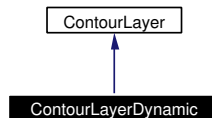
The documentation for this class was generated from the following file:

- ContourChannel.C

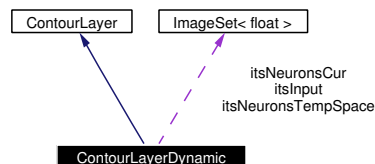
## 10.61 ContourLayerDynamic Class Reference

Based on Achim Braun's unpublished Neural Computation paper.

Inheritance diagram for ContourLayerDynamic:



Collaboration diagram for ContourLayerDynamic:



### Public Member Functions

- **ContourLayerDynamic** (const [ImageSet< float >](#) &imageMap, [SharedPtr< ParamMap >](#) pmap, const [ContourConnection](#) &connection)
- virtual [Image< float >](#) **iterate** (const [ContourConnection](#) &connection, const char \*savePrefix, const char \*saveSuffix, [Raster::NormalizeMode](#) normMode, bool doSave)

#### 10.61.1 Detailed Description

Based on Achim Braun's unpublished Neural Computation paper.

Definition at line 619 of file ContourChannel.C.

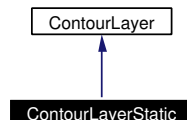
The documentation for this class was generated from the following file:

- ContourChannel.C

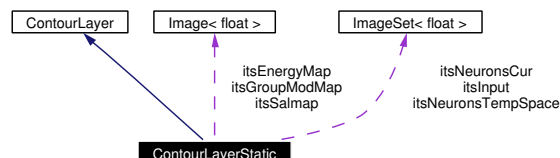
## 10.62 ContourLayerStatic Class Reference

Based on [CINNIC](#) (Nathan Mundhenk).

Inheritance diagram for ContourLayerStatic:



Collaboration diagram for ContourLayerStatic:



### Public Member Functions

- **ContourLayerStatic** (const [ImageSet](#)< float > &imageMap, [SharedPtr](#)< [ParamMap](#) > pmap, const [ContourConnection](#) &connection)
- virtual [Image](#)< float > **iterate** (const [ContourConnection](#) &connection, const char \*savePrefix, const char \*saveSuffix, [Raster::NormalizeMode](#) normMode, bool doSave)

#### 10.62.1 Detailed Description

Based on [CINNIC](#) (Nathan Mundhenk).

Definition at line 813 of file `ContourChannel.C`.

The documentation for this class was generated from the following file:

- `ContourChannel.C`

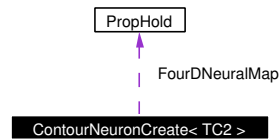


## 10.63 ContourNeuronCreate< TC2 > Class Template Reference

This class will create the neuron connection template.

```
#include <contourNeuron.H>
```

Collaboration diagram for ContourNeuronCreate< TC2 >:



### Public Member Functions

- float **FindRatio** (TC2 &XMax, TC2 &X)  
*find the ratio of XMax to X from 0 to 1*
- float **FindDistance** (TC2 &PiX, TC2 &PiY)  
*find the length from center to coord X,Y*
- void **FindCenter** ()  
*Calculate the where the center pixel is.*
- void **FindReach** ()  
*calculate the reach of excitation or inhabitation*
- float **FindAngleToRad** (TC2 &X, TC2 &Y)  
*find the angle to X,Y from the Center*
- float **FindRadToDeg** (TC2 &r)  
*Transform angle from radians to degrees.*
- float **FindLinear** (TC2 &m, TC2 &n, TC2 &o, TC2 &x, TC2 &b)  
*Convert along linear function  $y=mx+b$ .*
- float **sigmoid** (TC2 &beta, TC2 &v)  
*common sigmoidal function*
- **PropHold GetCompleteNeuron** ()

*Return the completed neuron.*

- void [CreateNeuron](#) ([readConfig](#) &config)

*Create the neuron using config file parameters.*

## Public Attributes

- [PropHold FourDNeuralMap](#) [AnglesUsed][AnglesUsed][XSize+1][YSize+1]

*create the 4D neuron map, return that map*

### 10.63.1 Detailed Description

**template<class TC2> class ContourNeuronCreate< TC2 >**

This class will create the neuron connection template.

This class creates the 4D neuron template. It processes the excitation and inhibition each neuron should have to another neuron given the property of the two neurons. The result is a set of weights. The weight for each hyperpixel is held in [PropHold](#). The entire set is held in a 4D array called FourDNeuralMap.

Definition at line 122 of file contourNeuron.H.

### 10.63.2 Member Function Documentation

**10.63.2.1 template<class TC2> void [ContourNeuronCreate](#)< TC2 >::CreateNeuron ([readConfig](#) & config)**

Create the neuron using config file parameters.

#### Parameters:

**&config** This is the config file object containing parameters

Definition at line 186 of file contourNeuron.C.

References [PropHold::ang](#), [PropHold::ang2](#), [PropHold::ang3](#), [PropHold::angAB](#), [PropHold::angABD](#), [PropHold::coLinear](#), [PropHold::dis](#), [PropHold::Edis](#), [ContourNeuronCreate< TC2 >::FindAngleToRad\(\)](#), [ContourNeuronCreate< TC2 >::FindCenter\(\)](#), [ContourNeuronCreate< TC2 >::FindDistance\(\)](#), [ContourNeuronCreate< TC2 >::FindLinear\(\)](#), [ContourNeuronCreate< TC2 >::FindRadToDeg\(\)](#), [ContourNeuronCreate< TC2 >::FindRatio\(\)](#), [ContourNeuronCreate< TC2 >::FindReach\(\)](#), [ContourNeuronCreate< TC2 >::FourDNeuralMap](#), [readConfig::getItemValueF\(\)](#),

NeuralAngles, PropHold::Odis, PropHold::orth, PropHold::pol, PropHold::Sdis, PropHold::sender, and PropHold::zero.

#### 10.63.2.2 `template<class TC2> float ContourNeuronCreate< TC2 >::FindAngleToRad (TC2 & X, TC2 & Y)`

find the angle to X,Y from the Center

Definition at line 100 of file contourNeuron.C.

References Y.

Referenced by ContourNeuronCreate< TC2 >::CreateNeuron().

#### 10.63.2.3 `template<class TC2> void ContourNeuronCreate< TC2 >::FindCenter ()`

Calculate the where the center pixel is.

Definition at line 83 of file contourNeuron.C.

Referenced by ContourNeuronCreate< TC2 >::CreateNeuron().

#### 10.63.2.4 `template<class TC2> float ContourNeuronCreate< TC2 >::FindDistance (TC2 & PiX, TC2 & PiY)`

find the length from center to coord X,Y

Definition at line 74 of file contourNeuron.C.

References sqrt().

Referenced by ContourNeuronCreate< TC2 >::CreateNeuron().

#### 10.63.2.5 `template<class TC2> float ContourNeuronCreate< TC2 >::FindLinear (TC2 & m, TC2 & n, TC2 & o, TC2 & x, TC2 & b)`

Convert along linear function  $y=mx+b$ .

Definition at line 165 of file contourNeuron.C.

Referenced by ContourNeuronCreate< TC2 >::CreateNeuron().

#### 10.63.2.6 `template<class TC2> float ContourNeuronCreate< TC2 >::FindRadToDeg (TC2 & r)`

Transform angle from radians to degrees.

Definition at line 157 of file contourNeuron.C.

Referenced by ContourNeuronCreate< TC2 >::CreateNeuron().

**10.63.2.7** `template<class TC2> float ContourNeuronCreate< TC2 >::FindRatio (TC2 & XMax, TC2 & X)`

find the ratio of XMax to X from 0 to 1

Definition at line 66 of file contourNeuron.C.

Referenced by ContourNeuronCreate< TC2 >::CreateNeuron().

**10.63.2.8** `template<class TC2> void ContourNeuronCreate< TC2 >::FindReach ()`

calculate the reach of excitation or inhibition

Definition at line 90 of file contourNeuron.C.

Referenced by ContourNeuronCreate< TC2 >::CreateNeuron().

**10.63.2.9** `template<class TC2> PropHold ContourNeuronCreate< TC2 >::GetCompleteNeuron ()`

Return the completed neuron.

Definition at line 178 of file contourNeuron.C.

References ContourNeuronCreate< TC2 >::FourDNeuralMap.

**10.63.2.10** `template<class TC2> float ContourNeuronCreate< TC2 >::sigmoid (TC2 & beta, TC2 & v)`

common sigmoidal function

Definition at line 171 of file contourNeuron.C.

### 10.63.3 Member Data Documentation

**10.63.3.1** `template<class TC2> PropHold ContourNeuronCreate< TC2 >::FourDNeuralMap[AnglesUsed][AnglesUsed][XSize+1][YSize+1]`

create the 4D neuron map, return that map

Definition at line 137 of file contourNeuron.H.

Referenced by CINNIC::convolveTest(), ContourNeuronCreate< TC2 >::CreateNeuron(), ContourNeuronCreate< TC2 >::GetCompleteNeuron(), and CINNIC::viewNeuronTemplate().

The documentation for this class was generated from the following files:

- [contourNeuron.H](#)
- [contourNeuron.C](#)

## 10.64 ContourNeuronProp< CH, X, Y > Class Template Reference

Holds the property for each neuron.

```
#include <contourNeuronProp.H>
```

Collaboration diagram for ContourNeuronProp< CH, X, Y >:

### Public Member Functions

- [ContourNeuronProp \(\)](#)
- CH & [getCharge \(\)](#)  
*return this neurons charge*
- CH & [getThreshold \(\)](#)  
*return this neurons threshold*
- CH & [getSupressionMod \(\)](#)  
*return this neurons supression modifier*
- void [setSupressionMod \(CH &sup\)](#)  
*set up supression for this neuron*
- void [setSupressionBool \(\)](#)  
*set holder to know if supression has been incremented*
- void [setSupressionThresh \(CH &thresh, CH &changeVal, CH &max\)](#)  
*set up supression threshold for this neuron and its weight augementation*
- void [ResetTempCharge \(\)](#)  
*reset the "temporary" charges of this neuron*
- void [ResetCharge \(\)](#)  
*reset base charge*
- void [ResetChargeAll \(\)](#)  
*reset all charges*
- void [setUpperLimit \(\)](#)  
*set neuron potential to upper limit*
- void [Charge \(CH &\\_charge, bool \\_pol, bool \\_sender\)](#)

*input a charge to this neuron, give your polarity so it can be sent on properly*

- void [ChargeSimple](#) (CH &\_charge)  
*input a charge to this neuron (potential) without polarity*
- CH & [DisCharge](#) (bool \_pol, bool \_sender, int a, int cascadeType)  
*output a charge, make sure it goes to the correct polarity*
- void [setThreshold](#) (CH &Threshold, CH &Resistance)  
*Tell the neuron its saturation level.*

## Protected Attributes

- float [threshold](#)  
*Current threshold for this neuron.*

### 10.64.1 Detailed Description

**template<class CH, class X, class Y> class ContourNeuronProp< CH, X, Y >**

Holds the property for each neuron.

this class is used to process and hold the membrane potential of each neuron. It keeps track of total potential as well as potential that is "up" or "down" that is, to simulate the effects of tranfering energy from neuron to another neuron it holds energy in either a positive or negative energy bank.

Definition at line 56 of file contourNeuronProp.H.

### 10.64.2 Constructor & Destructor Documentation

**10.64.2.1 template<class CH, class X, class Y> [ContourNeuronProp](#)< CH, X, Y >::[ContourNeuronProp](#) ()**

This method when called will place a charge in this neuron telling it where the charge came from and the polarity properties of the placing neuron

Definition at line 57 of file contourNeuronProp.C.

### 10.64.3 Member Function Documentation

**10.64.3.1** `template<class CH, class X, class Y> void ContourNeuronProp<CH, X, Y>::Charge (CH & _charge, bool _pol, bool _sender) [inline]`

input a charge to this neuron, give your polarity so it can be sent on properly

**Parameters:**

***charge*** The charge to give this neuron

***\_pol*** the polarity you have

***\_sender*** Is true if you are a sender of cascade

Definition at line 105 of file `contourNeuronProp.C`.

**10.64.3.2** `template<class CH, class X, class Y> void ContourNeuronProp<CH, X, Y>::ChargeSimple (CH & _charge)`

input a charge to this neuron (potential) without polarity

**Parameters:**

***charge*** The charge to give this neuron

Definition at line 127 of file `contourNeuronProp.C`.

**10.64.3.3** `template<class CH, class X, class Y> CH & ContourNeuronProp<CH, X, Y>::DisCharge (bool _pol, bool _sender, int a, int cascadeType) [inline]`

output a charge, make sure it goes to the correct polarity

**Parameters:**

***\_pol*** Your polarity

***\_sender*** Are you a sender?

Definition at line 136 of file `contourNeuronProp.C`.

**10.64.3.4** `template<class CH, class X, class Y> CH & ContourNeuronProp<CH, X, Y>::getCharge ()`

return this neurons charge

Definition at line 67 of file `contourNeuronProp.C`.



**10.64.3.5** `template<class CH, class X, class Y> CH & ContourNeuronProp<CH, X, Y >::getSupressionMod ()`

return this neurons supression modifier

Definition at line 79 of file contourNeuronProp.C.

**10.64.3.6** `template<class CH, class X, class Y> CH & ContourNeuronProp<CH, X, Y >::getThreshold ()`

return this neurons threshold

Definition at line 73 of file contourNeuronProp.C.

References ContourNeuronProp< CH, X, Y >::threshold.

**10.64.3.7** `template<class CH, class X, class Y> void ContourNeuronProp<CH, X, Y >::ResetCharge ()`

reset base charge

Definition at line 184 of file contourNeuronProp.C.

**10.64.3.8** `template<class CH, class X, class Y> void ContourNeuronProp<CH, X, Y >::ResetChargeAll ()`

reset all charges

Definition at line 190 of file contourNeuronProp.C.

**10.64.3.9** `template<class CH, class X, class Y> void ContourNeuronProp<CH, X, Y >::ResetTempCharge ()`

reset the "temporary" charges of this neuron

This method resets the place counter for the charge storage vector in effect putting its charge to 0;

Definition at line 177 of file contourNeuronProp.C.

**10.64.3.10** `template<class CH, class X, class Y> void ContourNeuronProp<CH, X, Y >::setSupressionBool ()`

set holder to know if supression has been incremented

Definition at line 91 of file contourNeuronProp.C.

**10.64.3.11** `template<class CH, class X, class Y> void ContourNeuronProp<CH, X, Y >::setSupressionMod (CH & sup)`

set up supression for this neuron

Definition at line 85 of file contourNeuronProp.C.

**10.64.3.12** `template<class CH, class X, class Y> void ContourNeuronProp<CH, X, Y >::setSupressionThresh (CH & thresh, CH & changeVal, CH & max)`

set up supression threshold for this neuron and its weight augementation

This method is used to set up adaptive supression for its threshold and the amount supression will increase every time threshold is reached

**Parameters:**

*thresh* This is the threshold number

*changeVal* The amount supression should be increased

Definition at line 97 of file contourNeuronProp.C.

**10.64.3.13** `template<class CH, class X, class Y> void ContourNeuronProp<CH, X, Y >::setThreshold (CH & Threshold, CH & Resistance)`

Tell the neuron its saturation level.

this method stores the neurons fireing threshold at which point it fires a salience charge

Definition at line 167 of file contourNeuronProp.C.

References ContourNeuronProp< CH, X, Y >::threshold.

**10.64.3.14** `template<class CH, class X, class Y> void ContourNeuronProp<CH, X, Y >::setUpperLimit ()`

set neuron potential to upper limit

Definition at line 198 of file contourNeuronProp.C.

## 10.64.4 Member Data Documentation

**10.64.4.1** `template<class CH, class X, class Y> float ContourNeuronProp<CH, X, Y >::threshold [protected]`

Current threshold for this neuron.

Definition at line 73 of file contourNeuronProp.H.

Referenced by `ContourNeuronProp< CH, X, Y >::getThreshold()`, and `ContourNeuronProp< CH, X, Y >::setThreshold()`.

The documentation for this class was generated from the following files:

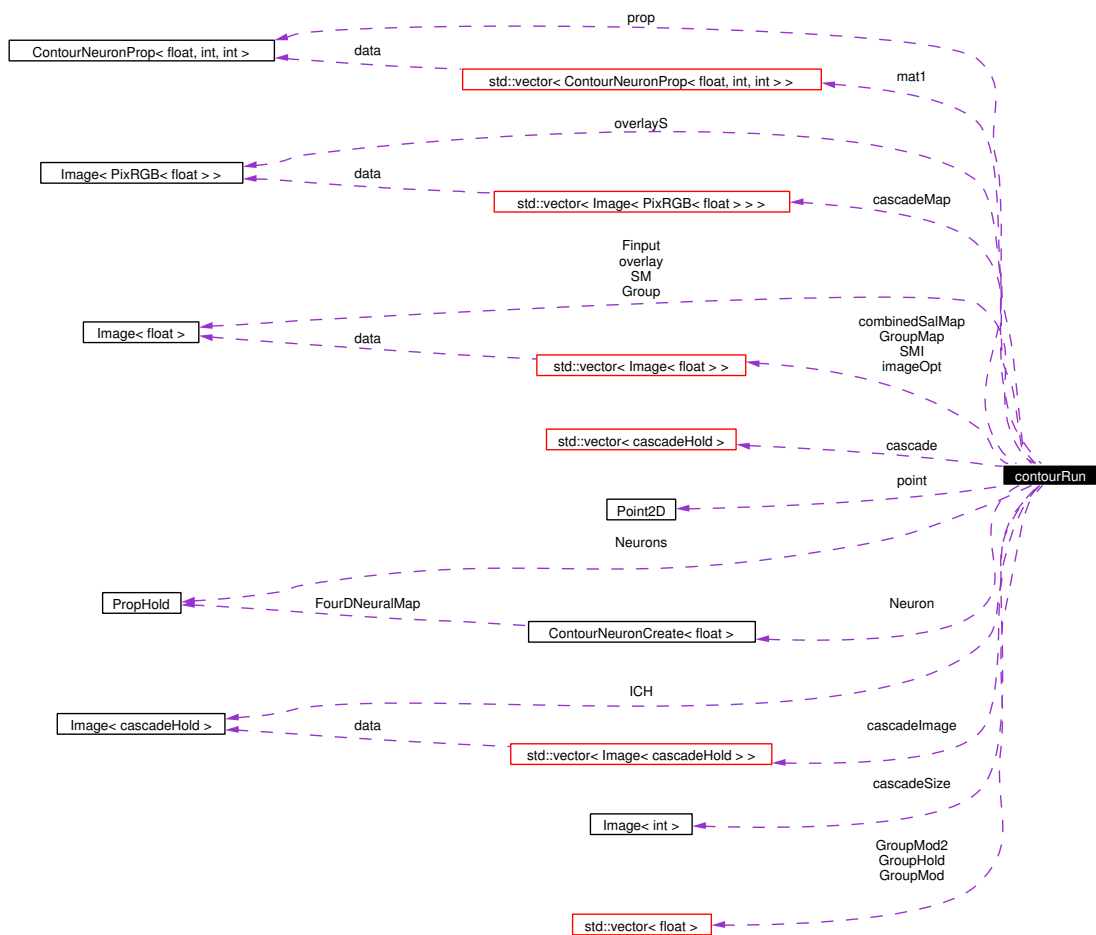
- [contourNeuronProp.H](#)
- [contourNeuronProp.C](#)

## 10.65 contourRun Class Reference

Run the hyper column on a given image with a given connection template.

```
#include <contourRun.H>
```

Collaboration diagram for contourRun:



### Public Member Functions

- void `copyCombinedSalMap` (`std::vector< Image< float > >` &CSM)  
*copy the combined sal map to this class for analysis or dumping*
- void `setArraySize` (long size)

*set the storage array size for linear components in subsq. iteration*

- [Image](#)< float > [getSMI](#) (int iter)

*returned the processed salmap for this object at iter*

- void [setIterations](#) (int iter)

*set number of iterations*

- void [deriveEnergy](#) ()

*find the time to total energy ratio*

- void [dumpEnergySigmoid](#) (const char \*filename, const char \*savefile, [readConfig](#) &config, [Image](#)< float > image, int scaleNo, int scaleTot)

*Dump energy values computed with sigmoidal process.*

- void [contourRunMain](#) (std::vector< [Image](#)< float > > &imageMap, [ContourNeuronCreate](#)< float > &N, [readConfig](#) &config, [Image](#)< float > &group, int groups, int iter, float groupTop)

*This is a short cut method to run runImage + other methods.*

## Public Attributes

- long [iterCounter](#)

### 10.65.1 Detailed Description

Run the hyper column on a given image with a given connection template.

This is the central iterative process to [CINNIC](#). It takes as input a 3D [ImageMap](#) and a 4D [PropHold](#) made from [ContourNeuronCreate](#). It matches each neuron in the hyper column against each other neuron in the hyper column. Energy is tranfered to the other neuron based on a product of the two neurons current exitation from the [ImageMap](#) and the weight of thier connections. A negative weight signifies inhabition. Energys are allowed to build iteratively in an upper layer SalMap built into this class.

Definition at line 94 of file contourRun.H.

## 10.65.2 Member Function Documentation

**10.65.2.1** `void contourRun::contourRunMain (std::vector< Image< float > > & imageMap, ContourNeuronCreate< float > & N, readConfig & config, Image< float > & group, int groups, int iter, float groupTop)`

This is a short cut method to run runImage + other methods.

This will run setConfig setImageSize, resetMatrix and runImage for you.

**Parameters:**

*Image* the processed image map of orientations from ImageMap

*N* This is the generic neuron template from ContourNeuronCreate

*config* This is the config file object from readConfig.C

*sizeX* The size of the image in X

*sizeY* The size of the image in Y

Definition at line 533 of file contourRun.C.

**10.65.2.2** `void contourRun::copyCombinedSalMap (std::vector< Image< float > > & CSM)`

copy the combined sal map to this class for analysis or dumping

Definition at line 76 of file contourRun.C.

**10.65.2.3** `void contourRun::deriveEnergy ()`

find the time to total energy ratio

find the max energy per time slice, timestep is 1/x seconds where x = timestep

Definition at line 216 of file contourRun.C.

**10.65.2.4** `void contourRun::dumpEnergySigmoid (const char * filename, const char * savefile, readConfig & config, Image< float > image, int scaleNo, int scaleTot)`

Dump energy values computed with sigmoidal process.

**Parameters:**

*filename* The base name to give this file, should include path

*savefile* The tailend of the file name

*readConfig* the config file object

Definition at line 144 of file contourRun.C.

References ASSERT, Image< byte >::getHeight(), Image< PixRGB< float > >::getHeight(), Image< byte >::getWidth(), Image< PixRGB< float > >::getWidth(), image, overlayStain(), rescale(), Image< T >::resize(), Image< T >::setVal(), and Raster::WriteRGB().

#### 10.65.2.5 [Image< float > contourRun::getSMI \(int iter\)](#)

returned the processed salmap for this object at iter

Definition at line 70 of file contourRun.C.

#### 10.65.2.6 [void contourRun::setArraySize \(long size\)](#)

set the storage array size for linear components in subsq. iteration

Definition at line 114 of file contourRun.C.

#### 10.65.2.7 [void contourRun::setIterations \(int iter\)](#)

set number of iterations

Definition at line 207 of file contourRun.C.

References ASSERT.

The documentation for this class was generated from the following files:

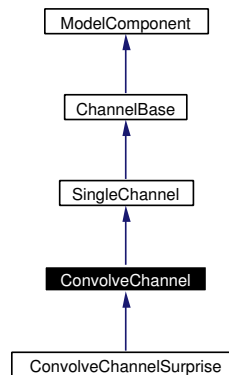
- [contourRun.H](#)
- [contourRun.C](#)

## 10.66 ConvolveChannel Class Reference

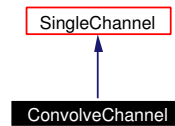
A generic convolution channel.

```
#include <Channels.H>
```

Inheritance diagram for ConvolveChannel:



Collaboration diagram for ConvolveChannel:



### Public Member Functions

- [ConvolveChannel](#) ([ModelManager](#) &mgr)  
*Construct with standard params. Will need to set a filter before using.*
- void [setFilter](#) (const [Image](#)< float > &filter)  
*Set the filter to use for convolutions.*
- virtual [~ConvolveChannel](#) ()  
*Destructor.*



### 10.66.1 Detailed Description

A generic convolution channel.

Definition at line 72 of file Channels.H.

### 10.66.2 Constructor & Destructor Documentation

#### 10.66.2.1 ConvolveChannel::ConvolveChannel ([ModelManager](#) & mgr)

Construct with standard params. Will need to set a filter before using.

Definition at line 90 of file Channels.C.

References [CONVOLVE](#), and [ModelParam< bool >::setVal\(\)](#).

#### 10.66.2.2 ConvolveChannel::~~ConvolveChannel () [virtual]

Destructor.

Definition at line 103 of file Channels.C.

### 10.66.3 Member Function Documentation

#### 10.66.3.1 void ConvolveChannel::setFilter (const [Image](#)< float > & filter)

Set the filter to use for convolutions.

Definition at line 98 of file Channels.C.

References [makeShared\(\)](#), and [SingleChannel::setPyramid\(\)](#).

The documentation for this class was generated from the following files:

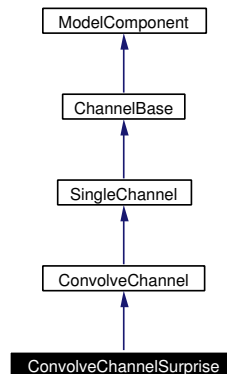
- [Channels.H](#)
- [Channels.C](#)

## 10.67 ConvolveChannelSurprise Class Reference

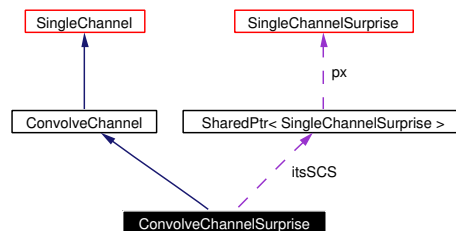
A generic convolution channel.

```
#include <ChannelsSurprise.H>
```

Inheritance diagram for ConvolveChannelSurprise:



Collaboration diagram for ConvolveChannelSurprise:



### Public Member Functions

- [ConvolveChannelSurprise](#) ([ModelManager](#) &mgr)  
*Construct with standard params. Will need to set a filter before using.*
- virtual [~ConvolveChannelSurprise](#) ()  
*Destructor.*
- virtual [Image](#)< float > [computeSubmap](#) (const [uint](#) index) const  
*[computeSubmap\(\)](#) is special and will return surprise*

## Protected Attributes

- [SharedPtr< SingleChannelSurprise > itsSCS](#)  
*our surprise computer*

### 10.67.1 Detailed Description

A generic convolution channel.

Definition at line 69 of file ChannelsSurprise.H.

### 10.67.2 Constructor & Destructor Documentation

#### 10.67.2.1 ConvolveChannelSurprise::ConvolveChannelSurprise ([ModelManager](#) & *mgr*)

Construct with standard params. Will need to set a filter before using.

Definition at line 79 of file ChannelsSurprise.C.

References [ModelComponent::addSubComponent\(\)](#).

#### 10.67.2.2 ConvolveChannelSurprise::~~ConvolveChannelSurprise () [virtual]

Destructor.

Definition at line 85 of file ChannelsSurprise.C.

### 10.67.3 Member Function Documentation

#### 10.67.3.1 [Image< float > ConvolveChannelSurprise::computeSubmap](#) (const [uint index](#)) const [virtual]

[computeSubmap\(\)](#) is special and will return surprise

Reimplemented from [SingleChannel](#).

Definition at line 88 of file ChannelsSurprise.C.

References [uint](#).

## 10.67.4 Member Data Documentation

### 10.67.4.1 [SharedPtr<SingleChannelSurprise> ConvolveChannelSurprise::its-SCS](#) [protected]

our surprise computer

Definition at line 82 of file ChannelsSurprise.H.

The documentation for this class was generated from the following files:

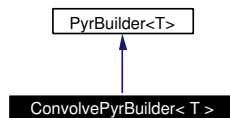
- [ChannelsSurprise.H](#)
- ChannelsSurprise.C

## 10.68 ConvolvePyrBuilder< T > Class Template Reference

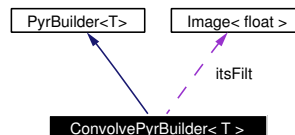
Builds pyramids based on arbitrary filters.

```
#include <PyrBuilder.H>
```

Inheritance diagram for ConvolvePyrBuilder< T >:



Collaboration diagram for ConvolvePyrBuilder< T >:



### Public Member Functions

- `ConvolvePyrBuilder` (const `Image< float >` &filt)  
*Constructor.*
- virtual `ConvolvePyrBuilder< T > * clone ()` const  
*Cloning constructor:.*
- virtual `ImageSet< T > build` (const `Image< T >` &img, const int depth)  
*Builds a pyramid based on img.*

#### 10.68.1 Detailed Description

```
template<class T> class ConvolvePyrBuilder< T >
```

Builds pyramids based on arbitrary filters.

Definition at line 115 of file `PyrBuilder.H`.

## 10.68.2 Constructor & Destructor Documentation

### 10.68.2.1 `template<class T> ConvolvePyrBuilder< T >::ConvolvePyrBuilder(const Image< float > & filt)`

Constructor.

Definition at line 88 of file `PyrBuilder.C`.

## 10.68.3 Member Function Documentation

### 10.68.3.1 `template<class T> ImageSet< T > ConvolvePyrBuilder< T >::build(const Image< T > & img, const int depth) [virtual]`

Builds a pyramid based on *img*.

Implements [PyrBuilder](#)< T >.

Definition at line 93 of file `PyrBuilder.C`.

References `buildPyrConvolve()`.

### 10.68.3.2 `template<class T> ConvolvePyrBuilder< T > * ConvolvePyrBuilder< T >::clone() const [virtual]`

Cloning constructor:.

Implements [PyrBuilder](#)< T >.

Definition at line 97 of file `PyrBuilder.C`.

The documentation for this class was generated from the following files:

- [PyrBuilder.H](#)
- [PyrBuilder.C](#)

## 10.69 CpuTimer Class Reference

Utility class for monitoring user/system cpu usage plus wall-clock time.

```
#include <CpuTimer.H>
```

### Public Member Functions

- [CpuTimer \(\)](#)  
*Constructor; immediately start timing.*
- void [reset \(\)](#)  
*Reset the timer and immediately start timing again.*
- void [mark \(\)](#)  
*Mark the current time for access via [user\\_secs\(\)](#)/[sys\\_secs\(\)](#)/[real\\_secs\(\)](#).*
- double [user\\_secs \(\)](#) const  
*Get the user CPU seconds between [reset\(\)](#) and [mark\(\)](#).*
- double [sys\\_secs \(\)](#) const  
*Get the system CPU seconds between [reset\(\)](#) and [mark\(\)](#).*
- double [real\\_secs \(\)](#) const  
*Get the real (wall-clock) CPU seconds between [reset\(\)](#) and [mark\(\)](#).*
- void [report \(const char \\*where=""\)](#) const  
*Print a report based on the most recent [mark\(\)](#).*

### 10.69.1 Detailed Description

Utility class for monitoring user/system cpu usage plus wall-clock time.

Definition at line 46 of file CpuTimer.H.

### 10.69.2 Constructor & Destructor Documentation

#### 10.69.2.1 CpuTimer::CpuTimer () [inline]

Constructor; immediately start timing.

Definition at line 79 of file CpuTimer.H.

References `reset()`.

### 10.69.3 Member Function Documentation

#### 10.69.3.1 `void CpuTimer::mark ()` [inline]

Mark the current time for access via `user_secs()`/`sys_secs()`/`real_secs()`.

Definition at line 91 of file `CpuTimer.H`.

#### 10.69.3.2 `double CpuTimer::real_secs () const` [inline]

Get the real (wall-clock) CPU seconds between `reset()` and `mark()`.

Definition at line 111 of file `CpuTimer.H`.

Referenced by `report()`.

#### 10.69.3.3 `void CpuTimer::report (const char * where = "") const` [inline]

Print a report based on the most recent `mark()`.

Definition at line 118 of file `CpuTimer.H`.

References `real_secs()`, `sys_secs()`, and `user_secs()`.

#### 10.69.3.4 `void CpuTimer::reset ()` [inline]

Reset the timer and immediately start timing again.

Definition at line 82 of file `CpuTimer.H`.

Referenced by `CpuTimer()`.

#### 10.69.3.5 `double CpuTimer::sys_secs () const` [inline]

Get the system CPU seconds between `reset()` and `mark()`.

Definition at line 104 of file `CpuTimer.H`.

Referenced by `report()`.

#### 10.69.3.6 `double CpuTimer::user_secs () const` [inline]

Get the user CPU seconds between `reset()` and `mark()`.



Definition at line 97 of file CpuTimer.H.

Referenced by report().

The documentation for this class was generated from the following file:

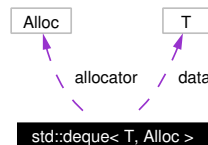
- CpuTimer.H

## 10.70 `std::deque< T, Alloc >` Class Template Reference

The STL deque container class. Use with `#include` .

`#include <dummySTL.H>`

Collaboration diagram for `std::deque< T, Alloc >`:



### Public Attributes

- `T data`
- `Alloc allocator`

#### 10.70.1 Detailed Description

`template<class T, class Alloc> class std::deque< T, Alloc >`

The STL deque container class. Use with `#include` .

See [SGI's documentation](#) of `std::deque`.

Definition at line 19 of file `dummySTL.H`.

The documentation for this class was generated from the following file:

- [dummySTL.H](#)

## 10.71 Dims Class Reference

A simple struct to hold a pair of width/height dimensions.

```
#include <Dims.H>
```

### Public Member Functions

- `Dims ()` throw ()  
*Default construct with width and height == 0.*
- `Dims (int width, int height)` throw ()  
*Construct from a given width + height, which must be non-negative.*
- `Dims (const Dims &other)` throw ()  
*Copy construct.*
- `int w ()` const throw ()  
*Returns the width, will always be non-negative.*
- `int h ()` const throw ()  
*Returns the height, will always be non-negative.*
- `int sz ()` const throw ()  
*Returns the size of the 2-D area specified by the dimensions.*
- `int max ()` const throw ()  
*Returns the larger of the width and height.*
- `int min ()` const throw ()  
*Returns the smaller of the width and height.*
- `bool isEmpty ()` const throw ()  
*Query whether the 2-D array is empty.*
- `bool isNonEmpty ()` const throw ()  
*Query whether the 2-D array is non-empty.*
- `bool operator== (const Dims &x)` const throw ()  
*See if two Dims are the same.*
- `bool operator!= (const Dims &x)` const throw ()  
*See if two Dims are different.*

### 10.71.1 Detailed Description

A simple struct to hold a pair of width/height dimensions.

We guarantee that the width and height are always non-negative, though not necessarily strictly positive. The default Dims are the 0-by-0 dimensions. Dims is strictly a const class; the width/height can only be set at construction.

Definition at line 47 of file Dims.H.

### 10.71.2 Constructor & Destructor Documentation

#### 10.71.2.1 Dims::Dims () throw () [inline]

Default construct with width and height == 0.

Definition at line 103 of file Dims.H.

#### 10.71.2.2 Dims::Dims (int width, int height) throw () [inline]

Construct from a given width + height, which must be non-negative.

Definition at line 108 of file Dims.H.

References ASSERT.

#### 10.71.2.3 Dims::Dims (const Dims & other) throw () [inline]

Copy construct.

Definition at line 115 of file Dims.H.

### 10.71.3 Member Function Documentation

#### 10.71.3.1 int Dims::h () const throw () [inline]

Returns the height, will always be non-negative.

Definition at line 124 of file Dims.H.

Referenced by Image< T >::attenuateBorders(), VisualBuffer::bufferToRetinal(), chamfer34(), SingleChannelBeoServer::check(), Point2D::clampToDims(), PsychoDisplay::clearScreen(), MonkeySaccadeController::computeWhenNewDecision(), FrictionSaccadeController::computeWhenNewDecision(), convertToString(), corrpach(), PsychoDisplay::createYUVoverlay(), PsychoDisplay::displayEyeTrackerCalibration(), PsychoDisplay::displayFixation(), PsychoDisplay::displayISCANcalib(), PsychoDisplay::displayText(), PsychoDisplay::displayYUVoverlay(),

RetinaStd::doGetOutput(), DirectFeedChannel::doInput(), downSize(), downSizeClean(), SimulationViewerStd::drawMotionSixPack(), TemplateMatchChannel::drawResults(), EyeTrackerSaccadeController::evolve(), SimulationViewerEyeMvt2::eyeMovement(), Rectangle::fitToDims(), Point2D::fitToDims(), PyrFoveator::foveate(), fromRGB(), fromYUV410P(), fromYUV411P(), fromYUV420P(), fromYUV422P(), fromYUV444P(), gaussianBlob(), MonkeySaccadeController::getEye(), MonkeySaccadeController::getHead(), FrameGrabber::getHeight(), InputFrameSeries::getImageDims(), VisualCortex::getJetDistMap(), MonkeySaccadeController::getJustEye(), VisualEvent::getMaxObjectDims(), VisualCortex::getOutput(), FlickerChannel::getOutput(), VisualCortex::getRaoJetMap(), SimulationViewerStd::getTraj(), SimulationViewerEyeMvt2::getTraj(), SimulationViewerEyeMvt::getTraj(), SimulationViewerCompress::getTraj(), infoFFT(), VisualBuffer::input(), isDyadic(), main(), PsychoDisplay::makeBlittableSurface(), omniCorrectGen(), ModelEyeHeadController::pix2deg(), MonkeyEyeHeadController::pix2deg(), Rectangle::Rectangle(), VisualBuffer::retinalToBuffer(), MbariResultViewer::saveSingleEventFrame(), VisualBuffer::start1(), V4Lgrabber::start1(), IEEE1394grabber::start1(), PsychoDisplay::start2(), InputFrameSeries::start2(), TaskRelevanceMapStd::update(), warp3D(), writeImageToStream(), and BitObject::writeToStream().

### 10.71.3.2 **bool Dims::isEmpty () const throw () [inline]**

Query whether the 2-D array is empty.

(empty means that at least one of the width or height is 0).

Definition at line 140 of file Dims.H.

Referenced by MbariResultViewer::displayImage(), RetinaStd::doGetOutput(), FrameSeries::resizeImage(), InputFrameSeries::start2(), and warp3D().

### 10.71.3.3 **bool Dims::isNonEmpty () const throw () [inline]**

Query whether the 2-D array is non-empty.

(i.e. both width and height are positive, non-zero).

Definition at line 144 of file Dims.H.

Referenced by downSizeClean(), Rectangle::fitToDims(), SimulationViewerStd::getTraj(), and Rectangle::modify().

### 10.71.3.4 **int Dims::max () const throw () [inline]**

Returns the larger of the width and height.

Definition at line 132 of file Dims.H.

**10.71.3.5 int Dims::min () const throw () [inline]**

Returns the smaller of the width and height.

Definition at line 136 of file Dims.H.

**10.71.3.6 bool Dims::operator!= (const Dims & x) const throw () [inline]**

See if two Dims are different.

Definition at line 151 of file Dims.H.

**10.71.3.7 bool Dims::operator== (const Dims & x) const throw () [inline]**

See if two Dims are the same.

Definition at line 147 of file Dims.H.

**10.71.3.8 int Dims::sz () const throw () [inline]**

Returns the size of the 2-D area specified by the dimensions.

(i.e. width \* height).

Definition at line 128 of file Dims.H.

Referenced by ArrayData< T >::ArrayData(), Image< T >::getSize(), and ArrayData< T >::~~ArrayData().

**10.71.3.9 int Dims::w () const throw () [inline]**

Returns the width, will always be non-negative.

Definition at line 120 of file Dims.H.

Referenced by Image< T >::attenuateBorders(), VisualBuffer::bufferToRetinal(), chamfer34(), SingleChannelBeoServer::check(), Point2D::clampToDims(), PsychoDisplay::clearScreen(), MonkeySaccadeController::computeWhenNewDecision(), FrictionSaccadeController::computeWhenNewDecision(), convertToString(), corrpach(), PsychoDisplay::createYUVoverlay(), PsychoDisplay::displayEyeTrackerCalibration(), PsychoDisplay::displayFixation(), PsychoDisplay::displayISCANcalib(), PsychoDisplay::displayText(), PsychoDisplay::displayYUVoverlay(), RetinaStd::doGetOutput(), DirectFeedChannel::doInput(), downSize(), downSizeClean(), SimulationViewerStd::drawMotionSixPack(), TemplateMatchChannel::drawResults(), EyeTrackerSaccadeController::evolve(), SimulationViewerEyeMvt2::eyeMovement(), Rectangle::fitToDims(), Point2D::fitToDims(), PyrFoveator::foveate(), fromRGB(), fromYUV410P(), fromYUV411P(), fromYUV420P(), fromYUV422P(),

fromYUV444P(), gaussianBlob(), MonkeySaccadeController::getEye(), MonkeySaccadeController::getHead(), InputFrameSeries::getImageDims(), VisualCortex::getJetDistMap(), MonkeySaccadeController::getJustEye(), VisualEvent::getMaxObjectDims(), SoxChannel::getNonlinearResponse(), VisualCortex::getOutput(), FlickerChannel::getOutput(), VisualCortex::getRaoJetMap(), SimulationViewerStd::getTraj(), SimulationViewerEyeMvt2::getTraj(), SimulationViewerEyeMvt::getTraj(), SimulationViewerCompress::getTraj(), FrameGrabber::getWidth(), infoFFT(), VisualBuffer::input(), isDyadic(), main(), PsychoDisplay::makeBlittableSurface(), omniCorrectGen(), ModelEyeHeadController::pix2deg(), MonkeyEyeHeadController::pix2deg(), SingleChannel::postProcessMap(), Rectangle::Rectangle(), VisualBuffer::retinalToBuffer(), MbariResultViewer::saveSingleEventFrame(), VisualBuffer::start1(), V4Lgrabber::start1(), IEEE1394grabber::start1(), PsychoDisplay::start2(), InputFrameSeries::start2(), TaskRelevanceMapStd::update(), warp3D(), writeImageToStream(), and BitObject::writeToStream().

The documentation for this class was generated from the following file:

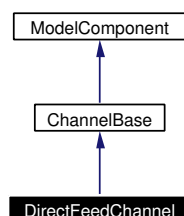
- [Dims.H](#)

## 10.72 DirectFeedChannel Class Reference

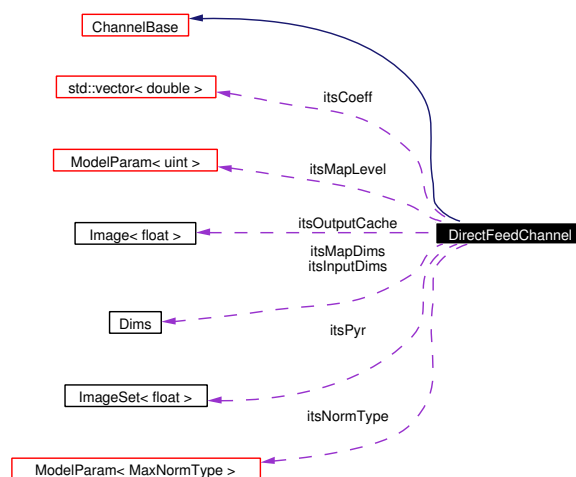
Direct Feed Channel.

```
#include <DirectFeedChannel.H>
```

Inheritance diagram for DirectFeedChannel:



Collaboration diagram for DirectFeedChannel:



### Public Member Functions

- [DirectFeedChannel](#) ([ModelManager](#) &mgr, int id=-1)

*Constructor.*

- virtual [~DirectFeedChannel](#) ()

*Destructor.*



- virtual void [exportOptions](#) (const int whatToExport, const bool recurse=true)  
*Export some of our [ModelParam](#) parameters as command-line-options.*
- virtual void [reset](#) (const bool recurse=true)  
*Reset [SingleChannel](#).*
- virtual void [readFrom](#) (const [ParamMap](#) &pmap)  
*Read params from the [ParamMap](#).*
- virtual void [writeTo](#) ([ParamMap](#) &pmap) const  
*Write params to the [ParamMap](#).*
- virtual void [cleanCoeffs](#) (const double eps)  
*Set any coefficients less than eps to zero.*
- virtual void [clampCoeffs](#) (const double cmin, const double cmax)  
*Clamp the weighting coefficients to the specified range.*
- virtual double [absSumCoeffs](#) () const  
*Return the sum of the absolute values of the weighting coefficients.*
- virtual void [setCoeff](#) (const [uint](#) idx, const double val)  
*set coefficient idx to val*
- virtual double [getCoeff](#) (const [uint](#) idx) const  
*returns coefficient idx*
- virtual void [normalizeCoeffs](#) (const double div)  
*Normalize the weighting coefficients by dividing by div.*
- void [initializeCoeffs](#) (const [uint](#) numCoeffs, const double val=1.0)  
*initialize the coefficients to value val*
- virtual void [learn](#) (const [Image](#)< [byte](#) > &dmap, const double eta, const bool softmax=false, const int inthresh=20, const int outthresh=255)  
*Adjust the weighting coefficients according to the target map.*
- virtual bool [outputAvailable](#) () const  
*Query whether the channel is ready to give yield valid output.*
- virtual const [Dims](#) & [getMapDims](#) () const  
*Return the dimensions of the output image.*

- virtual const [Dims](#) & [getInputDims](#) () const  
*Return the dimensions of the most recent input image.*
- virtual void [fillJet](#) (const [Point2D](#) &loc, [Jet](#)< float > &j, [std::vector](#)< int > &index) const  
*Fill the components of the given [Jet](#) that relate to this channel.*
- virtual void [fillJetNI](#) (const [Point2D](#) &loc, [Jet](#)< float > &j, [std::vector](#)< int > &index) const  
*Fill the components of the given [Jet](#) that relate to this channel.*
- virtual void [fillJetSpec](#) ([JetSpec](#) &js) const  
*Fill the specification of a [Jet](#) with data that relates to this channel.*
- virtual [uint](#) [numSubmaps](#) () const  
*Return the number of feature maps represented by this channel.*
- virtual [Image](#)< float > [getSubmap](#) (const [uint](#) index) const  
*Get the feature map at the given index.*
- virtual const char \* [getSubmapName](#) (const [uint](#) index) const  
*Get the name/description of the feature map at the given index.*
- virtual void [biasFeatures](#) (const double \*mean, const double \*sigma, int &idx)  
*not implemented*
- virtual void [getFeatures](#) (const [Point2D](#) &locn, [std::vector](#)< double > &mean) const  
*not implemented*
- virtual void [findWeight](#) ()  
*not implemented*
- virtual bool [reco](#) (const [ParamMap](#) &pmap, double &mismatch)  
*not implemented*
- virtual void [findLikelihood](#) (const [ParamMap](#) &pmap, double &likelihood)  
*not implemented*
- virtual [Image](#)< float > [getOutput](#) ()  
*The output will be the size of the feature map level in the pyramid.*

- virtual void [setListener](#) ([SharedPtr](#)< [ChannelListener](#) > listener)  
*not implemented*
- virtual [uint](#) [numChans](#) () const  
*number of subchannels - always returns 1*
- virtual void [killCaches](#) ()  
*erase all internal caches*
- virtual void [inputPyramid](#) (const [ImageSet](#)< float > &pyramid, const double t)  
*Send an already computed pyramid as input to the channel.*

### Protected Member Functions

- virtual void [doInput](#) (const [Image](#)< [PixRGB](#)< float > > \*colimg, const [Image](#)< float > \*bwimg, const double t, const [Image](#)< [byte](#) > &clipMask)  
*the actual input routine*
- virtual void [computeOutput](#) ()  
*computed the output of the channel*
- virtual [Image](#)< float > [applyMaxNorm](#) ([Image](#)< float > &map)  
*applies the max norm to map depending on itsNormType*

### Protected Attributes

- [ModelParam](#)< [uint](#) > [itsMapLevel](#)  
*the level at which the saliency map lives*
- [ModelParam](#)< [MaxNormType](#) > [itsNormType](#)  
*the kind of normalization to be applied*

#### 10.72.1 Detailed Description

Direct Feed Channel.

Definition at line 47 of file DirectFeedChannel.H.

## 10.72.2 Constructor & Destructor Documentation

### 10.72.2.1 `DirectFeedChannel::DirectFeedChannel (ModelManager & mgr, int id = -1)`

Constructor.

**Parameters:**

*id* if given, becomes part of the name of the channel

Definition at line 44 of file `DirectFeedChannel.C`.

References `DIRECTFEED`, `MaxNormType`, `ModelComponent::setDescriptiveName()`, `ModelComponent::setTagName()`, `toStr()`, and `VCXNORM_FANCY`.

### 10.72.2.2 `DirectFeedChannel::~DirectFeedChannel ()` [virtual]

Destructor.

Definition at line 59 of file `DirectFeedChannel.C`.

## 10.72.3 Member Function Documentation

### 10.72.3.1 `double DirectFeedChannel::absSumCoeffs () const` [virtual]

Return the sum of the absolute values of the weighting coefficients.

Implements [ChannelBase](#).

Definition at line 121 of file `DirectFeedChannel.C`.

References `abs()`, and `uint`.

### 10.72.3.2 `Image< float > DirectFeedChannel::applyMaxNorm (Image< float > & map)` [protected, virtual]

applies the max norm to map depending on its `NormType`

Definition at line 334 of file `DirectFeedChannel.C`.

References `ASSERT`, `ModelParam< MaxNormType >::getVal()`, `ModelParam< float >::getVal()`, `Image< T >::initialized()`, and `maxNormalize()`.

Referenced by `computeOutput()`.

### 10.72.3.3 void DirectFeedChannel::biasFeatures (const double \* *mean*, const double \* *sigma*, int & *idx*) [virtual]

not implemented

Implements [ChannelBase](#).

Definition at line 270 of file DirectFeedChannel.C.

### 10.72.3.4 void DirectFeedChannel::clampCoeffs (const double *cmin*, const double *cmax*) [virtual]

Clamp the weighting coefficients to the specified range.

Implements [ChannelBase](#).

Definition at line 113 of file DirectFeedChannel.C.

References CLAMP(), killCaches(), and uint.

### 10.72.3.5 void DirectFeedChannel::cleanCoeffs (const double *eps*) [virtual]

Set any coefficients less than eps to zero.

Implements [ChannelBase](#).

Definition at line 105 of file DirectFeedChannel.C.

References killCaches(), and uint.

### 10.72.3.6 void DirectFeedChannel::computeOutput () [protected, virtual]

computed the output of the channel

Definition at line 318 of file DirectFeedChannel.C.

References applyMaxNorm(), ASSERT, getMapDims(), rescale(), ImageSet< float >::size(), uint, and ZEROS.

Referenced by getOutput().

### 10.72.3.7 void DirectFeedChannel::doInput (const [Image](#)< [PixRGB](#)< float > > \* *colimg*, const [Image](#)< float > \* *bwimg*, const double *t*, const [Image](#)< byte > & *clipMask*) [protected, virtual]

the actual input routine

Only `colimg` or `bwing` are needed to obtain the dimensions. After each call of `doinput`, `inputPyramid` must be called.

Implements [ChannelBase](#).

Definition at line 185 of file `DirectFeedChannel.C`.

References `Image< T >::getDims()`, `ModelParam< uint >::getVal()`, `Dims::h()`, `itsMapLevel`, `toStr()`, and `Dims::w()`.

#### 10.72.3.8 `void DirectFeedChannel::exportOptions (const int whatToExport, const bool recurse = true) [virtual]`

Export some of our [ModelParam](#) parameters as command-line-options.

See the base function in [ModelComponent.H](#) for info.

Reimplemented from [ChannelBase](#).

Definition at line 63 of file `DirectFeedChannel.C`.

References `ChannelBase::exportOptions()`, `itsMapLevel`, `OPTEXP_CORE`, and `ModelManager::requestOption()`.

#### 10.72.3.9 `void DirectFeedChannel::fillJet (const Point2D & loc, Jet< float > & j, std::vector< int > & index) const [virtual]`

Fill the components of the given [Jet](#) that relate to this channel.

All relevant components of the [Jet](#) are set to this channel's values at the specified location across all pyramid levels. The channel will store its data starting at the given index in the jet, and will return the number of entries filled. The index vector will be modified as the various [Jet](#) components are explored, but will be restored to its original state upon completion.

Implements [ChannelBase](#).

Definition at line 227 of file `DirectFeedChannel.C`.

#### 10.72.3.10 `void DirectFeedChannel::fillJetNI (const Point2D & loc, Jet< float > & j, std::vector< int > & index) const [virtual]`

Fill the components of the given [Jet](#) that relate to this channel.

This version does not attempt to use interpolation

Implements [ChannelBase](#).

Definition at line 235 of file `DirectFeedChannel.C`.

**10.72.3.11** `void DirectFeedChannel::fillJetSpec (JetSpec & js) const`  
[virtual]

Fill the specification of a [Jet](#) with data that relates to this channel.

Implements [ChannelBase](#).

Definition at line 243 of file DirectFeedChannel.C.

**10.72.3.12** `void DirectFeedChannel::findLikelihood (const ParamMap & pmap,`  
`double & likelihood) [virtual]`

not implemented

Implements [ChannelBase](#).

Definition at line 301 of file DirectFeedChannel.C.

**10.72.3.13** `void DirectFeedChannel::findWeight () [virtual]`

not implemented

Implements [ChannelBase](#).

Definition at line 286 of file DirectFeedChannel.C.

**10.72.3.14** `double DirectFeedChannel::getCoeff (const uint idx) const`  
[virtual]

returns coefficient idx

Definition at line 148 of file DirectFeedChannel.C.

References ASSERT, numSubmaps(), and uint.

**10.72.3.15** `void DirectFeedChannel::getFeatures (const Point2D & locn,`  
`std::vector< double > & mean) const [virtual]`

not implemented

Implements [ChannelBase](#).

Definition at line 278 of file DirectFeedChannel.C.

**10.72.3.16** `const Dims & DirectFeedChannel::getInputDims () const`  
`[virtual]`

Return the dimensions of the most recent input image.

Implements [ChannelBase](#).

Definition at line 223 of file `DirectFeedChannel.C`.

**10.72.3.17** `const Dims & DirectFeedChannel::getMapDims () const`  
`[virtual]`

Return the dimensions of the output image.

Implements [ChannelBase](#).

Definition at line 219 of file `DirectFeedChannel.C`.

Referenced by `computeOutput()`.

**10.72.3.18** `Image< float > DirectFeedChannel::getOutput ()` `[virtual]`

The output will be the size of the feature map level in the pyramid.

(I.e., the size of the output is given by [getMapDims\(\)](#)).

Implements [ChannelBase](#).

Definition at line 309 of file `DirectFeedChannel.C`.

References `ASSERT`, `computeOutput()`, and `Image< float >::initialized()`.

Referenced by `MultiDirectFeedChannel::combineOutputs()`.

**10.72.3.19** `Image< float > DirectFeedChannel::getSubmap (const uint index)`  
`const` `[virtual]`

Get the feature map at the given index.

This function is intended to be implemented by subclasses so that it recurses into any nested channels, mapping nested submaps into a linear sequence of indices.

Implements [ChannelBase](#).

Definition at line 254 of file `DirectFeedChannel.C`.

References `ASSERT`, `numSubmaps()`, and `uint`.



**10.72.3.20** `const char * DirectFeedChannel::getSubmapName (const uint index) const` `[virtual]`

Get the name/description of the feature map at the given index.

This function is intended to be implemented by subclasses so that it recurses into any nested channels, mapping nested submaps into a linear sequence of indices. The output of this function is intended only to be human-readable (e.g. for labeling image displays), and does not have to follow any particular parseable syntax.

Implements [ChannelBase](#).

Definition at line 261 of file DirectFeedChannel.C.

References [ASSERT](#), [ModelComponent::descriptiveName\(\)](#), [numSubmaps\(\)](#), and [uint](#).

**10.72.3.21** `void DirectFeedChannel::initializeCoeffs (const uint numCoeffs, const double val = 1.0)`

initialize the coefficients to value val

Definition at line 155 of file DirectFeedChannel.C.

References [uint](#).

Referenced by [inputPyramid\(\)](#), and [setCoeff\(\)](#).

**10.72.3.22** `void DirectFeedChannel::inputPyramid (const ImageSet< float > & pyramid, const double t)` `[virtual]`

Send an already computed pyramid as input to the channel.

Must be called after each call of input.

Definition at line 172 of file DirectFeedChannel.C.

References [ASSERT](#), [initializeCoeffs\(\)](#), and [ImageSet< T >::size\(\)](#).

Referenced by [MultiDirectFeedChannel::inputPyramid\(\)](#).

**10.72.3.23** `void DirectFeedChannel::killCaches ()` `[virtual]`

erase all internal caches

Reimplemented from [ChannelBase](#).

Definition at line 355 of file DirectFeedChannel.C.

References [Image< float >::freeMem\(\)](#).

Referenced by `clampCoeffs()`, `cleanCoeffs()`, `normalizeCoeffs()`, and `setCoeff()`.

**10.72.3.24** `void DirectFeedChannel::learn (const Image< byte > & dmap, const double eta, const bool softmax = false, const int inthresh = 20, const int outhresh = 255) [virtual]`

Adjust the weighting coefficients according to the target map.

Implements [ChannelBase](#).

Definition at line 162 of file `DirectFeedChannel.C`.

**10.72.3.25** `void DirectFeedChannel::normalizeCoeffs (const double div) [virtual]`

Normalize the weighting coefficients by dividing by *div*.

Implements [ChannelBase](#).

Definition at line 130 of file `DirectFeedChannel.C`.

References `killCaches()`, and `uint`.

**10.72.3.26** `uint DirectFeedChannel::numChans () const [virtual]`

number of subchannels - always returns 1

Implements [ChannelBase](#).

Definition at line 351 of file `DirectFeedChannel.C`.

References `uint`.

**10.72.3.27** `uint DirectFeedChannel::numSubmaps () const [virtual]`

Return the number of feature maps represented by this channel.

Implements [ChannelBase](#).

Definition at line 250 of file `DirectFeedChannel.C`.

References `ImageSet< float >::size()`, and `uint`.

Referenced by `getCoeff()`, `getSubmap()`, `getSubmapName()`, and `setCoeff()`.

**10.72.3.28** `bool DirectFeedChannel::outputAvailable () const [virtual]`

Query whether the channel is ready to give yield valid output.

In some cases (e.g. with motion channels), it may take several input frames before the channel is fully initialized.

Implements [ChannelBase](#).

Definition at line 212 of file DirectFeedChannel.C.

#### 10.72.3.29 void DirectFeedChannel::readFrom (const [ParamMap](#) & *pmap*) [virtual]

Read params from the [ParamMap](#).

Reimplemented from [ChannelBase](#).

Definition at line 91 of file DirectFeedChannel.C.

References [ChannelBase::readFrom\(\)](#).

#### 10.72.3.30 bool DirectFeedChannel::reco (const [ParamMap](#) & *pmap*, double & *mismatch*) [virtual]

not implemented

Implements [ChannelBase](#).

Definition at line 293 of file DirectFeedChannel.C.

#### 10.72.3.31 void DirectFeedChannel::reset (const bool *recurse* = true) [virtual]

Reset [SingleChannel](#).

See the base function in [ModelComponent.H](#) for info.

Reimplemented from [ModelComponent](#).

Definition at line 77 of file DirectFeedChannel.C.

References [Image< float >::freeMem\(\)](#), [ModelComponent::reset\(\)](#), and [ImageSet< float >::reset\(\)](#).

#### 10.72.3.32 void DirectFeedChannel::setCoeff (const [uint](#) *idx*, const double *val*) [virtual]

set coefficient idx to val

Definition at line 138 of file DirectFeedChannel.C.

References [ASSERT](#), [initializeCoeffs\(\)](#), [ImageSet< float >::isEmpty\(\)](#), [killCaches\(\)](#), [numSubmaps\(\)](#), and [uint](#).

**10.72.3.33** `void DirectFeedChannel::setListener (SharedPtr< ChannelListener  
> listener) [virtual]`

not implemented

Implements [ChannelBase](#).

Definition at line 344 of file DirectFeedChannel.C.

**10.72.3.34** `void DirectFeedChannel::writeTo (ParamMap & pmap) const  
[virtual]`

Write params to the [ParamMap](#).

Reimplemented from [ChannelBase](#).

Definition at line 98 of file DirectFeedChannel.C.

References [ChannelBase::writeTo\(\)](#).

## 10.72.4 Member Data Documentation

**10.72.4.1** `ModelParam<uint> DirectFeedChannel::itsMapLevel  
[protected]`

the level at which the saliency map lives

Definition at line 197 of file DirectFeedChannel.H.

Referenced by [doInput\(\)](#), and [exportOptions\(\)](#).

**10.72.4.2** `ModelParam<MaxNormType> DirectFeedChannel::itsNormType  
[protected]`

the kind of normalization to be applied

Definition at line 200 of file DirectFeedChannel.H.

The documentation for this class was generated from the following files:

- [DirectFeedChannel.H](#)
- [DirectFeedChannel.C](#)

## 10.73 DirectionChannel Class Reference

a motion sensitive channel with direction selectivity

```
#include <Channels.H>
```

Inheritance diagram for DirectionChannel: Collaboration diagram for DirectionChannel:

### Public Member Functions

- [DirectionChannel](#) ([ModelManager](#) &mgr, const [uint](#) dirIndex, const double direction, const [PyramidType](#) type)  
*Constructor with some parameters.*
- virtual void [exportOptions](#) (const int whatToExport, const bool recurse=true)  
*Export some of our [ModelParam](#) parameters as command-line-options.*
- virtual [~DirectionChannel](#) ()  
*Destructor.*

### Protected Member Functions

- virtual void [doInput](#) (const [Image](#)< [PixRGB](#)< float > > \*, const [Image](#)< float > \*bwimg, const double t, const [Image](#)< [byte](#) > &clipMask)  
*ReichardtChannel requires only luminance input.*

### Protected Attributes

- [ModelParam](#)< [uint](#) > **itsIndex**
- [ModelParam](#)< double > **itsDirection**
- [ModelParam](#)< bool > **itsTakeSqrt**

#### 10.73.1 Detailed Description

a motion sensitive channel with direction selectivity

Definition at line 333 of file Channels.H.

## 10.73.2 Constructor & Destructor Documentation

### 10.73.2.1 DirectionChannel::DirectionChannel ([ModelManager](#) & mgr, const [uint](#) dirIndex, const double direction, const [PyramidType](#) type)

Constructor with some parameteres.

**Parameters:**

**dirIndex** is the channel's index, used in Param and [Jet](#) access

**direction** is the preferred direction in degrees (0..360)

**type** the type of the pyramid used for computing the Reichardt motion.

Definition at line 1406 of file Channels.C.

References [makeShared\(\)](#), [ModelComponent::setDescriptiveName\(\)](#), [ModelComponent::setTagName\(\)](#), [ModelParam< float >::setVal\(\)](#), [ModelParam< bool >::setVal\(\)](#), and [uint](#).

### 10.73.2.2 DirectionChannel::~~DirectionChannel () [virtual]

Destructor.

Definition at line 1445 of file Channels.C.

## 10.73.3 Member Function Documentation

### 10.73.3.1 void DirectionChannel::doInput (const [Image](#)< [PixRGB](#)< float > > \*, const [Image](#)< float > \* bwing, const double t, const [Image](#)< byte > & clipMask) [protected, virtual]

ReichardtChannel requires only luminance input.

After sending the input image to the ReichardtPyramid, rectify and threshold the Reichardt maps.

Reimplemented from [SingleChannel](#).

Reimplemented in [DirectionChannelBeo](#).

Definition at line 1449 of file Channels.C.

References [SingleChannel::doInput\(\)](#), [doSqrt\(\)](#), [ModelParam< bool >::getVal\(\)](#), and [SingleChannel::pyrMut\(\)](#).

### 10.73.3.2 void DirectionChannel::exportOptions (const int *whatToExport*, const bool *recurse* = true) [virtual]

Export some of our [ModelParam](#) parameters as command-line-options.

See the base function in [ModelComponent.H](#) for info.

Reimplemented from [SingleChannel](#).

Definition at line 1433 of file Channels.C.

References [SingleChannel::exportOptions\(\)](#), `OPTEXP_CORE`, and [Model-Manager::requestOption\(\)](#).

The documentation for this class was generated from the following files:

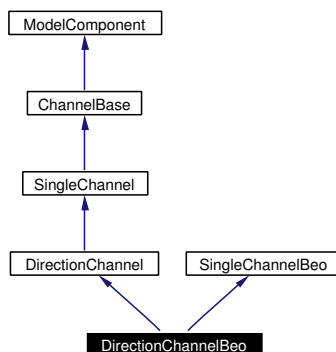
- [Channels.H](#)
- [Channels.C](#)

## 10.74 DirectionChannelBeo Class Reference

Beowulf version of [DirectionChannel](#).

```
#include <ChannelsBeo.H>
```

Inheritance diagram for DirectionChannelBeo:



Collaboration diagram for DirectionChannelBeo:

### Public Member Functions

- **DirectionChannelBeo** ([ModelManager](#) &mgr, const [uint](#) dirIndex, const double direction, const [PyramidType](#) type, const [SharedPtr](#)< [Beowulf](#) > &beow)
- virtual bool [outputAvailable](#) () const

*Query whether the channel is ready to give yield valid output.*

### Protected Member Functions

- virtual void [doInput](#) (const [Image](#)< [PixRGB](#)< float > > \*, const [Image](#)< float > > \*bwimg, const double t, const [Image](#)< byte > &clipMask)
- virtual void [start1](#) ()

*ReichardtChannel requires only luminance input.*

#### 10.74.1 Detailed Description

Beowulf version of [DirectionChannel](#).

Definition at line 144 of file ChannelsBeo.H.



## 10.74.2 Member Function Documentation

**10.74.2.1** `void DirectionChannelBeo::doInput (const Image< PixRGB< float > > *, const Image< float > * bwimg, const double t, const Image< byte > & clipMask)` [protected, virtual]

ReichardtChannel requires only luminance input.

After sending the input image to the ReichardtPyramid, rectify and threshold the Reichardt maps.

Reimplemented from [DirectionChannel](#).

Definition at line 225 of file ChannelsBeo.C.

References [SingleChannelBeo::sendInput\(\)](#).

**10.74.2.2** `bool DirectionChannelBeo::outputAvailable () const` [virtual]

Query whether the channel is ready to give yield valid output.

In some cases (e.g. with motion channels), it may take several input frames before the channel is fully initialized.

Reimplemented from [SingleChannel](#).

Definition at line 218 of file ChannelsBeo.C.

References [SingleChannel::outputAvailable\(\)](#).

**10.74.2.3** `void DirectionChannelBeo::start1 ()` [protected, virtual]

If you overload this, make sure you call [SingleChannel::start1\(\)](#) at the beginning of your overload

Reimplemented from [SingleChannel](#).

Definition at line 229 of file ChannelsBeo.C.

References [SingleChannelBeo::requestBeowulfNode\(\)](#), and [SingleChannel::start1\(\)](#).

The documentation for this class was generated from the following files:

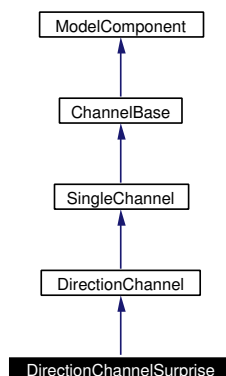
- [ChannelsBeo.H](#)
- [ChannelsBeo.C](#)

## 10.75 DirectionChannelSurprise Class Reference

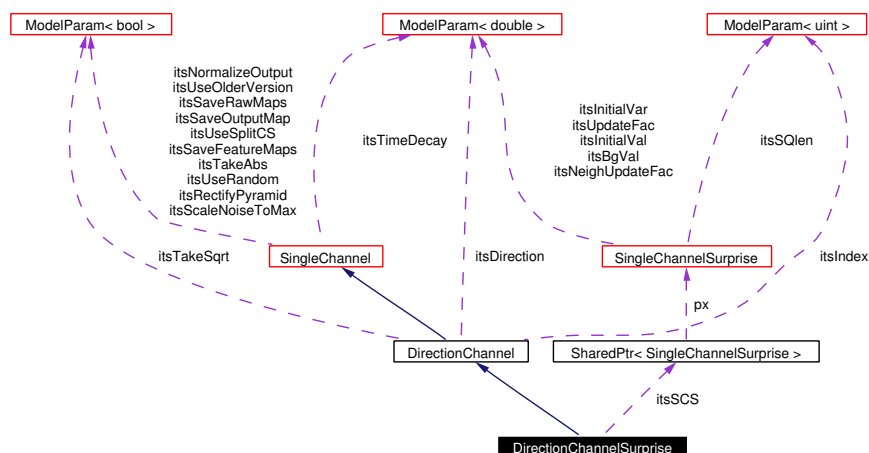
a motion sensitive channel with direction selectivity

```
#include <ChannelsSurprise.H>
```

Inheritance diagram for DirectionChannelSurprise:



Collaboration diagram for DirectionChannelSurprise:



### Public Member Functions

- **DirectionChannelSurprise** (**ModelManager** &mgr, const **uint** dirIndex, const double direction, const **PyramidType** type)

Constructor with some parameters.

- virtual [~DirectionChannelSurprise](#) ()

Destructor.

- virtual [Image](#)< float > [computeSubmap](#) (const [uint](#) index) const  
*[computeSubmap\(\)](#) is special and will return surprise*

## Protected Attributes

- [SharedPtr](#)< [SingleChannelSurprise](#) > [itsSCS](#)  
*our surprise computer*

### 10.75.1 Detailed Description

a motion sensitive channel with direction selectivity

Definition at line 322 of file ChannelsSurprise.H.

### 10.75.2 Constructor & Destructor Documentation

#### 10.75.2.1 DirectionChannelSurprise::DirectionChannelSurprise ([ModelManager](#) & *mgr*, const [uint](#) *dirIndex*, const double *direction*, const [PyramidType](#) *type*)

Constructor with some parameters.

Definition at line 292 of file ChannelsSurprise.C.

References [ModelComponent::addSubComponent\(\)](#), and [uint](#).

#### 10.75.2.2 DirectionChannelSurprise::~~DirectionChannelSurprise () [virtual]

Destructor.

Definition at line 301 of file ChannelsSurprise.C.

### 10.75.3 Member Function Documentation

#### 10.75.3.1 [Image](#)< float > [DirectionChannelSurprise::computeSubmap](#) (const [uint](#) *index*) const [virtual]

[computeSubmap\(\)](#) is special and will return surprise

Reimplemented from [SingleChannel](#).

Definition at line 304 of file ChannelsSurprise.C.

References [uint](#).

### 10.75.4 Member Data Documentation

#### 10.75.4.1 [SharedPtr](#)<[SingleChannelSurprise](#)> [DirectionChannelSurprise::its-SCS](#) [protected]

our surprise computer

Definition at line 336 of file ChannelsSurprise.H.

The documentation for this class was generated from the following files:

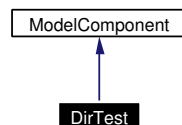
- [ChannelsSurprise.H](#)
- ChannelsSurprise.C

## 10.76 DirTest Class Reference

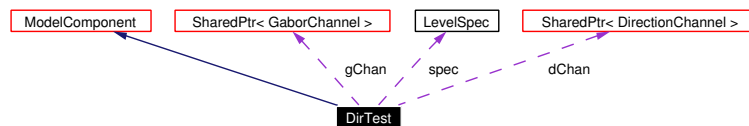
Class for testing [DirectionChannel](#).

```
#include <DirTest.H>
```

Inheritance diagram for DirTest:



Collaboration diagram for DirTest:



### Public Types

- enum { **DISP\_GABOR** = 0, **DISP\_DIR** = 1 }

### Public Member Functions

- **DirTest** ([ModelManager](#) &mgr, int dir, int numDirs=4)
- [Image< byte >](#) **loadImage** (string fName)
- void **inputChan** ([Image< byte >](#) img)
- void **dispMinMaxAvg** ([Image< float >](#) img)
- void **dispGPstats** ()
- void **dispGPmaps** ()
- void **dispRstats** ()
- void **dispRmaps** ()
- void **dispSstats** ()
- void **dispSmaps** ()
- void **dispCstats** ()
- void **dispGmap** ()
- void **dispDmap** ()
- void **normalize** (bool norm)

- void **noRet** (bool ret)
- [SharedPtr< GaborChannel >](#) **gabChan** ()
- [SharedPtr< DirectionChannel >](#) **dirChan** ()

### 10.76.1 Detailed Description

Class for testing [DirectionChannel](#).

This is a test class providing a nice interface for testing the [DirectionChannel](#). It constructs the [DirectionChannel](#) in its constructor and provides methods to load images, input the images, display the output maps, display statistics for the maps, etc.

Definition at line 31 of file DirTest.H.

The documentation for this class was generated from the following files:

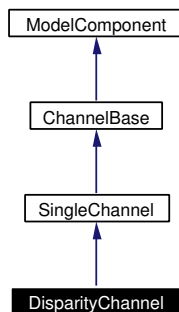
- [DirTest.H](#)
- [DirTest.C](#)

## 10.77 DisparityChannel Class Reference

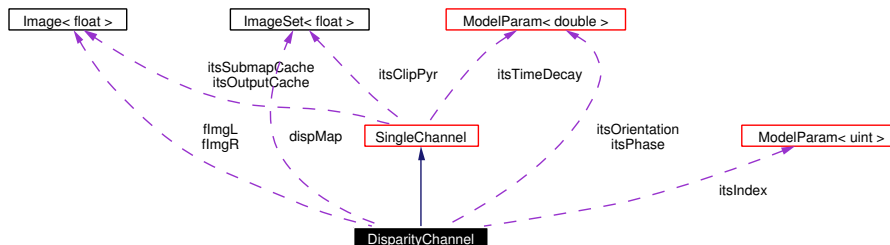
An disparity channel.

```
#include <Channels.H>
```

Inheritance diagram for DisparityChannel:



Collaboration diagram for DisparityChannel:



### Public Member Functions

- [DisparityChannel](#) ([ModelManager](#) &mgr, const [uint](#) dIndex, const double ang, const double pha)  
*Construct with standard params plus a disparity index for this object.*
- virtual [~DisparityChannel](#) ()  
*Destructor.*
- virtual void [exportOptions](#) (const int whatToExport, const bool recurse=true)  
*Export some of our [ModelParam](#) parameters as command-line-options.*

- double [angle](#) () const  
*Get the preferred orientation (in degrees) of this channel.*
- double [phase](#) () const  
*Get the preferred phase (in degrees) of this channel.*
- void [setRawFilteredImages](#) ([Image](#)< float > \*\*\*fImgLE, [Image](#)< float > \*\*\*f-  
ImgRI)  
*set the raw filtered images from stereo channel to avoid redundancy*
- void [getRawFilteredImages](#) ([Image](#)< float > \*\*\*\*fImgLE, [Image](#)< float >   
\*\*\*\*fImgRI)  
*access to the raw filtered images for easy viewing*
- void [getDispMap](#) ([ImageSet](#)< float > \*dMap)  
*access to the disparity even before stored in the queue ([SingleChannel](#))*
- void [normalizeDispMap](#) ([ImageSet](#)< float > tDispMap, int nChan)  
*normalize the disparity map to [0, 1]*
- void [setNumPhase](#) (int n)  
*to get the dimensions of the raw filtered images from stereo channel*
- void [setNumTheta](#) (int t)
- void [doInput](#) (const [Image](#)< [PixRGB](#)< float > > \*, const [Image](#)< float >   
\*bwimg, const double t, const [Image](#)< [byte](#) > &clipMask)  
*perform the disparity calculation*
- void [storePyramid](#) (double t)
- void [start1](#) ()  
*overload [start1\(\)](#)*

## Protected Attributes

- [ModelParam](#)< uint > **itsIndex**
- [ModelParam](#)< double > **itsOrientation**
- [ModelParam](#)< double > **itsPhase**



### 10.77.1 Detailed Description

An disparity channel.

Definition at line 711 of file Channels.H.

### 10.77.2 Constructor & Destructor Documentation

#### 10.77.2.1 DisparityChannel::DisparityChannel ([ModelManager](#) & *mgr*, const [uint](#) *dIndex*, const double *ang*, const double *pha*)

Construct with standard params plus a disparity index for this object.

**Parameters:**

*dIndex* is used for various indexing such as with [ParamMap](#) and [Jet](#) interfacing.

*ang* is the preferred orientation disparity for this channel, in degrees (0..180).

*pha* is the preferred phase disparity for this channel, in degrees (0..360).

Definition at line 521 of file Channels.C.

References [DISPARITY](#), [ModelComponent::setDescriptiveName\(\)](#), [ModelComponent::setTagName\(\)](#), and [uint](#).

#### 10.77.2.2 DisparityChannel::~DisparityChannel () [virtual]

Destructor.

Definition at line 540 of file Channels.C.

### 10.77.3 Member Function Documentation

#### 10.77.3.1 double DisparityChannel::angle () const

Get the preferred orientation (in degrees) of this channel.

Definition at line 565 of file Channels.C.

References [ModelParam< double >::getVal\(\)](#).

#### 10.77.3.2 void DisparityChannel::doInput (const [Image](#)< [PixRGB](#)< float > > \*, const [Image](#)< float > \* *bwimg*, const double *t*, const [Image](#)< byte > & *clipMask*) [virtual]

perform the disparity calculation

Reimplemented from [SingleChannel](#).

Definition at line 631 of file Channels.C.

References [buildPyrGaussian\(\)](#), [ModelParam< double >::getVal\(\)](#), [Image< float >::getWidth\(\)](#), [Image< T >::rectify\(\)](#), [shiftImage\(\)](#), [squared\(\)](#), [SingleChannel::storePyramid\(\)](#), [uint](#), and [ZEROS](#).

Referenced by [StereoChannel::doInput\(\)](#).

### 10.77.3.3 **void DisparityChannel::exportOptions (const int *whatToExport*, const bool *recurse* = true) [virtual]**

Export some of our [ModelParam](#) parameters as command-line-options.

See the base function in [ModelComponent.H](#) for info.

Reimplemented from [SingleChannel](#).

Definition at line 544 of file Channels.C.

References [SingleChannel::exportOptions\(\)](#).

### 10.77.3.4 **void DisparityChannel::getDispMap ([ImageSet< float > \\* dMap](#))**

access to the disparity even before stored in the queue ([SingleChannel](#))

Definition at line 593 of file Channels.C.

### 10.77.3.5 **void DisparityChannel::getRawFilteredImages ([Image< float > \\*\\*\\*\\* fImgLE](#), [Image< float > \\*\\*\\*\\* fImgRI](#))**

access to the raw filtered images for easy viewing

Definition at line 585 of file Channels.C.

### 10.77.3.6 **void DisparityChannel::normalizeDispMap ([ImageSet< float > tDispMap](#), int *nChan*)**

normalize the disparity map to [0, 1]

Definition at line 599 of file Channels.C.

References [uint](#).

### 10.77.3.7 **double DisparityChannel::phase () const**

Get the preferred phase (in degrees) of this channel.

Definition at line 571 of file Channels.C.

References `ModelParam< double >::getVal()`.

#### 10.77.3.8 void DisparityChannel::setNumPhase (int *n*)

to get the dimensions of the raw filtered images from stereo channel

Definition at line 619 of file Channels.C.

Referenced by `StereoChannel::buildSubChans()`.

#### 10.77.3.9 void DisparityChannel::setRawFilteredImages ([Image](#)< float > \*\*\* *fImgLE*, [Image](#)< float > \*\*\* *fImgRI*)

set the raw filtered images from stereo channel to avoid redundancy

Definition at line 577 of file Channels.C.

Referenced by `StereoChannel::doInput()`.

#### 10.77.3.10 void DisparityChannel::start1 () [virtual]

overload [start1\(\)](#)

Reimplemented from [SingleChannel](#).

Definition at line 554 of file Channels.C.

References `ModelParam< LevelSpec >::getVal()`, `LevelSpec::maxDepth()`, `ImageSet< float >::reset()`, and `SingleChannel::start1()`.

The documentation for this class was generated from the following files:

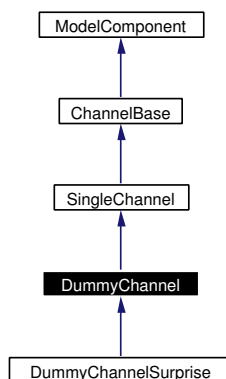
- [Channels.H](#)
- [Channels.C](#)

## 10.78 DummyChannel Class Reference

A dummy channel that simply passes its greyscale input to its output.

```
#include <Channels.H>
```

Inheritance diagram for DummyChannel:



Collaboration diagram for DummyChannel:

### Public Member Functions

- [DummyChannel](#) ([ModelManager](#) &mgr)  
*Construct with standard params.*
- virtual [~DummyChannel](#) ()  
*Destructor.*
- virtual void [exportOptions](#) (const int whatToExport, const bool recurse=true)  
*Export some of our [ModelParam](#) parameters as command-line-options.*
- bool [outputAvailable](#) () const  
*overload*
- const [Dims](#) & [getMapDims](#) () const  
*overload*

## Protected Member Functions

- virtual void `doInput` (const `Image< PixRGB< float > > *`, const `Image< float > *bwimg`, const double `t`, const `Image< byte > &clipMask`)  
*special input function*
- virtual `Image< float > centerSurround` (const `uint` `cntrlev`, const `uint` `surrlev`) const  
*special submap computation function*

## Protected Attributes

- `ModelParam< float > itsFactor`  
*output = input \* factor if non-zero*
- `Image< float > itsMap`  
*our output map*

### 10.78.1 Detailed Description

A dummy channel that simply passes its greyscale input to its output.

USE WITH CAUTION: only tested for `-levelspec=0-0,0-0,0`

Definition at line 643 of file `Channels.H`.

### 10.78.2 Constructor & Destructor Documentation

#### 10.78.2.1 `DummyChannel::DummyChannel (ModelManager & mgr)`

Construct with standard params.

Definition at line 2889 of file `Channels.C`.

#### 10.78.2.2 `DummyChannel::~~DummyChannel ()` [virtual]

Destructor.

Definition at line 2897 of file `Channels.C`.

### 10.78.3 Member Function Documentation

#### 10.78.3.1 **Image< float > DummyChannel::centerSurround (const uint *cntrlev*, const uint *surrlev*) const** [protected, virtual]

special submap computation function

Reimplemented from [SingleChannel](#).

Definition at line 2929 of file Channels.C.

References `ASSERT`, `LevelSpec::csOK()`, `ModelParam< LevelSpec >::getVal()`, and `uint`.

#### 10.78.3.2 **void DummyChannel::doInput (const Image< PixRGB< float > > \*, const Image< float > \* *bwimg*, const double *t*, const Image< byte > & *clipMask*)** [protected, virtual]

special input function

Reimplemented from [SingleChannel](#).

Definition at line 2920 of file Channels.C.

References `ModelParam< float >::getVal()`, and `itsFactor`.

#### 10.78.3.3 **void DummyChannel::exportOptions (const int *whatToExport*, const bool *recurse* = true)** [virtual]

Export some of our [ModelParam](#) parameters as command-line-options.

See the base function in [ModelComponent.H](#) for info.

Reimplemented from [SingleChannel](#).

Definition at line 2901 of file Channels.C.

References `SingleChannel::exportOptions()`, `itsFactor`, `OPTEXP_CORE`, and `ModelManager::requestOption()`.

#### 10.78.3.4 **const Dims & DummyChannel::getMapDims () const** [virtual]

overload

Reimplemented from [SingleChannel](#).

Definition at line 2916 of file Channels.C.

References `Image< float >::getDims()`.

**10.78.3.5** `bool DummyChannel::outputAvailable () const` [virtual]

overload

Reimplemented from [SingleChannel](#).

Definition at line 2912 of file Channels.C.

References `Image< float >::initialized()`.

**10.78.4 Member Data Documentation****10.78.4.1** `ModelParam<float> DummyChannel::itsFactor` [protected]

output = input \* factor if non-zero

Definition at line 664 of file Channels.H.

Referenced by `doInput()`, and `exportOptions()`.

**10.78.4.2** `Image<float> DummyChannel::itsMap` [protected]

our output map

Definition at line 675 of file Channels.H.

The documentation for this class was generated from the following files:

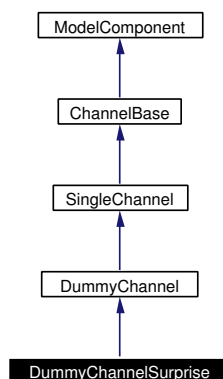
- [Channels.H](#)
- [Channels.C](#)

## 10.79 DummyChannelSurprise Class Reference

A dummy channel.

```
#include <ChannelsSurprise.H>
```

Inheritance diagram for DummyChannelSurprise:



Collaboration diagram for DummyChannelSurprise:

### Public Member Functions

- [DummyChannelSurprise](#) ([ModelManager](#) &mgr)  
*Constructor.*
- virtual [~DummyChannelSurprise](#) ()  
*Destructor.*
- virtual [Image](#)< float > [computeSubmap](#) (const [uint](#) index) const  
*[computeSubmap\(\)](#) is special and will return surprise*

### Protected Attributes

- [SharedPtr](#)< [SingleChannelSurprise](#) > itsSCS  
*our surprise computer*



### 10.79.1 Detailed Description

A dummy channel.

Definition at line 400 of file ChannelsSurprise.H.

### 10.79.2 Constructor & Destructor Documentation

#### 10.79.2.1 DummyChannelSurprise::DummyChannelSurprise ([ModelManager](#) & mgr)

Constructor.

Definition at line 364 of file ChannelsSurprise.C.

References [ModelComponent::addSubComponent\(\)](#).

#### 10.79.2.2 DummyChannelSurprise::~~DummyChannelSurprise () [virtual]

Destructor.

Definition at line 369 of file ChannelsSurprise.C.

### 10.79.3 Member Function Documentation

#### 10.79.3.1 [Image](#)< float > DummyChannelSurprise::computeSubmap (const [uint](#) index) const [virtual]

[computeSubmap\(\)](#) is special and will return surprise

Reimplemented from [SingleChannel](#).

Definition at line 372 of file ChannelsSurprise.C.

References [uint](#).

### 10.79.4 Member Data Documentation

#### 10.79.4.1 [SharedPtr](#)<[SingleChannelSurprise](#)> [DummyChannelSurprise::its-SCS](#) [protected]

our surprise computer

Definition at line 413 of file ChannelsSurprise.H.

The documentation for this class was generated from the following files:

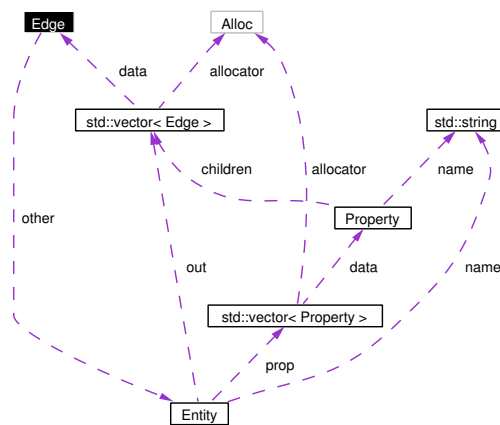
- [ChannelsSurprise.H](#)
- ChannelsSurprise.C

## 10.80 Edge Class Reference

Definition of Edge class and its functions.

```
#include <Graph.H>
```

Collaboration diagram for Edge:



### Public Member Functions

- [Edge \(\)](#)  
*Constructor.*
- [~Edge \(\)](#)  
*Destructor.*
- void [set\\_other](#) ([Entity](#) \*other)
- void [set\\_co\\_occur](#) (float c)  
*set co-occurrence*
- const [Entity](#) \* [get\\_other](#) () const  
*Get the destination vertex.*
- float [get\\_co\\_occur](#) () const  
*get co-occurrence*
- [Entity](#) \* [get\\_other\\_mutable](#) ()  
*get mutable access to destination vertex*

- float [find\\_error](#) ([Edge](#) ke)  
*find error btw known edge ke and this edge*
- void [setAll](#) (float d, float a, float b)  
*set distance, alpha and beta*

### 10.80.1 Detailed Description

Definition of Edge class and its functions.

Each Edge contains 2 fields: one field is a static measure of granularity and the second field is a dynamic quantity which measures the probability of co-occurrence of the 2 entities that this edge connects. To optimise, we removed the static quantity from the Edge and defined it as GRANULARITYx.

Definition at line 222 of file Graph.H.

### 10.80.2 Constructor & Destructor Documentation

#### 10.80.2.1 [Edge::Edge](#) () [inline]

Constructor.

Definition at line 428 of file Graph.H.

#### 10.80.2.2 [Edge::~~Edge](#) () [inline]

Destructor.

Definition at line 432 of file Graph.H.

### 10.80.3 Member Function Documentation

#### 10.80.3.1 float [Edge::find\\_error](#) ([Edge](#) ke)

find error btw known edge ke and this edge

Definition at line 88 of file Landmark\_Graph.C.

References [alpha](#), [beta](#), [distance](#), and [SQR\(\)](#).

**10.80.3.2** `float Edge::get_co_occur () const [inline]`

get co-occurrence

Definition at line 448 of file Graph.H.

**10.80.3.3** `const Entity * Edge::get_other () const [inline]`

Get the destination vertex.

Definition at line 444 of file Graph.H.

**10.80.3.4** `Entity * Edge::get_other_mutable () [inline]`

get mutable access to destination vertex

Definition at line 452 of file Graph.H.

**10.80.3.5** `void Edge::set_co_occur (float c) [inline]`

set co-occurrence

**Parameters:**

*co-occurrence* enter the co-occurrence of the entities related by this edge

Definition at line 440 of file Graph.H.

**10.80.3.6** `void Edge::set_other (Entity * other) [inline]`

set the destination vertex

**Parameters:**

*other* enter the other entity (destination vertex)

Definition at line 436 of file Graph.H.

**10.80.3.7** `void Edge::setAll (float d, float a, float b) [inline]`

set distance, alpha and beta

Definition at line 115 of file Landmark\_Graph.H.

The documentation for this class was generated from the following files:

- [Graph.H](#)

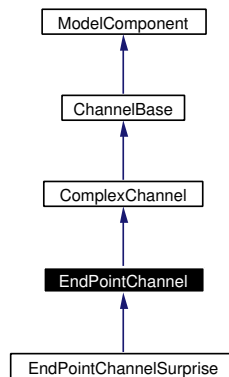
- [Landmark\\_Graph.H](#)
- [Landmark\\_Graph.C](#)

## 10.81 EndPointChannel Class Reference

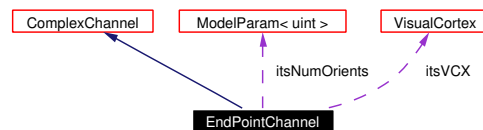
An end point detector channel that detects line ends at different.

```
#include <Channels.H>
```

Inheritance diagram for EndPointChannel:



Collaboration diagram for EndPointChannel:



### Public Member Functions

- [EndPointChannel](#) ([ModelManager](#) &mgr, [VisualCortex](#) \*vcx)  
*Construct with standard params.*
- virtual [~EndPointChannel](#) ()  
*Destructor.*
- virtual void [exportOptions](#) (const int whatToExport, const bool recurse=true)  
*Export some of our [ModelParam](#) parameters as command-line-options.*
- virtual bool [setModelParamString](#) (const [std::string](#) &name, const [std::string](#) &value, const bool recurse=false, const bool warn=true)

*Overload so that we can reconfigure when our params get changed.*

## Protected Member Functions

- virtual void `buildSubChans()`  
*(re-)build our subchannels*
- virtual void `doInput` (const `Image< PixRGB< float > > *`, const `Image< float > *bwimg`, const double `t`, const `Image< byte > &clipMask`)  
*Computes the end stops at different orientations.*

## Protected Attributes

- `ModelParam< uint > itsNumOrients`  
*number of junction subchannels*
- int `itsExportOpts`  
*keep track of what options to export in case we rebuild our subchans*
- `VisualCortex * itsVCX`

### 10.81.1 Detailed Description

An end point detector channel that detects line ends at different.

Definition at line 1405 of file Channels.H.

### 10.81.2 Constructor & Destructor Documentation

#### 10.81.2.1 `EndPointChannel::EndPointChannel (ModelManager & mgr, VisualCortex * vcx)`

Construct with standard params.

Definition at line 2105 of file Channels.C.

References `buildSubChans()`, `ENDPOINT`, and `OPTEXP_NONE`.



**10.81.2.2 EndPointChannel::~~EndPointChannel () [virtual]**

Destructor.

Definition at line 2116 of file Channels.C.

**10.81.3 Member Function Documentation****10.81.3.1 void EndPointChannel::buildSubChans () [protected, virtual]**

(re-)build our subchannels

Reimplemented in [EndPointChannelSurprise](#).

Definition at line 2120 of file Channels.C.

References [ComplexChannel::addSubChan\(\)](#), [ModelParam< uint >::getVal\(\)](#), [makeShared\(\)](#), [ModelComponent::removeAllSubComponents\(\)](#), [ChannelBase::setHomogeneous\(\)](#), [uint](#), and [ChannelBase::visualFeature\(\)](#).

Referenced by [EndPointChannel\(\)](#), and [setModelParamString\(\)](#).

**10.81.3.2 void EndPointChannel::doInput (const [Image](#)< [PixRGB](#)< float > > \*, const [Image](#)< float > \* *bwimg*, const double *t*, const [Image](#)< byte > & *clipMask*) [protected, virtual]**

Computes the end stops at different orientations.

Implements [ChannelBase](#).

Definition at line 2212 of file Channels.C.

References [ASSERT](#), [VisualCortex::hasChannel\(\)](#), [ChannelBase::input\(\)](#), [ComplexChannel::numChans\(\)](#), [ComplexChannel::subChan\(\)](#), and [uint](#).

**10.81.3.3 void EndPointChannel::exportOptions (const int *whatToExport*, const bool *recurse* = true) [virtual]**

Export some of our [ModelParam](#) parameters as command-line-options.

See the base function in [ModelComponent.H](#) for info.

Reimplemented from [ComplexChannel](#).

Definition at line 2197 of file Channels.C.

References [ComplexChannel::exportOptions\(\)](#), [OPTEXP\\_CORE](#), and [ModelManager::requestOption\(\)](#).

**10.81.3.4** `bool EndPointChannel::setModelParamString (const std::string & name, const std::string & value, const bool recurse = false, const bool warn = true) [virtual]`

Overload so that we can reconfigure when our params get changed.

Reimplemented from [ModelComponent](#).

Definition at line 2179 of file Channels.C.

References [buildSubChans\(\)](#), [ModelParamBase::getName\(\)](#), [ModelParam< uint >::getVal\(\)](#), [ComplexChannel::numChans\(\)](#), and [ModelComponent::setModelParamString\(\)](#).

## 10.81.4 Member Data Documentation

**10.81.4.1** `int EndPointChannel::itsExportOpts [protected]`

keep track of what options to export in case we rebuild our subchans

Definition at line 1437 of file Channels.H.

**10.81.4.2** `ModelParam<uint> EndPointChannel::itsNumOrients [protected]`

number of junction subchannels

Definition at line 1434 of file Channels.H.

The documentation for this class was generated from the following files:

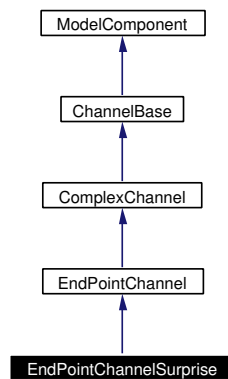
- [Channels.H](#)
- [Channels.C](#)

## 10.82 EndPointChannelSurprise Class Reference

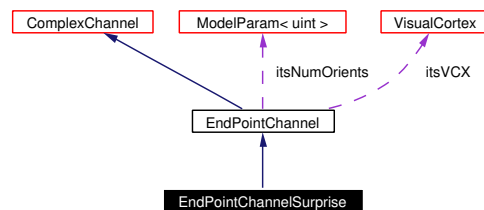
An end point detector channel that detects line ends at different.

```
#include <ChannelsSurprise.H>
```

Inheritance diagram for EndPointChannelSurprise:



Collaboration diagram for EndPointChannelSurprise:



### Public Member Functions

- [EndPointChannelSurprise](#) ([ModelManager](#) &mgr, [VisualCortex](#) \*vcx)

*Construct with standard params.*

- virtual [~EndPointChannelSurprise](#) ()

*Destructor.*

## Protected Member Functions

- virtual void [buildSubChans](#) ()  
(re-)build our subchannels

### 10.82.1 Detailed Description

An end point detector channel that detects line ends at different.

Definition at line 613 of file ChannelsSurprise.H.

### 10.82.2 Constructor & Destructor Documentation

#### 10.82.2.1 EndPointChannelSurprise::EndPointChannelSurprise ([ModelManager](#) & mgr, [VisualCortex](#) \* vcx)

Construct with standard params.

Definition at line 874 of file ChannelsSurprise.C.

References [buildSubChans\(\)](#).

#### 10.82.2.2 EndPointChannelSurprise::~~EndPointChannelSurprise () [virtual]

Destructor.

Definition at line 882 of file ChannelsSurprise.C.

### 10.82.3 Member Function Documentation

#### 10.82.3.1 void EndPointChannelSurprise::buildSubChans () [protected, virtual]

(re-)build our subchannels

Reimplemented from [EndPointChannel](#).

Definition at line 886 of file ChannelsSurprise.C.

References [ComplexChannel::addSubChan\(\)](#), [ModelParam< uint >::getVal\(\)](#), [makeShared\(\)](#), [ModelComponent::removeAllSubComponents\(\)](#), [ChannelBase::setHomogeneous\(\)](#), [uint](#), and [ChannelBase::visualFeature\(\)](#).

Referenced by [EndPointChannelSurprise\(\)](#).

The documentation for this class was generated from the following files:

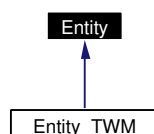
- [ChannelsSurprise.H](#)
- ChannelsSurprise.C

## 10.83 Entity Class Reference

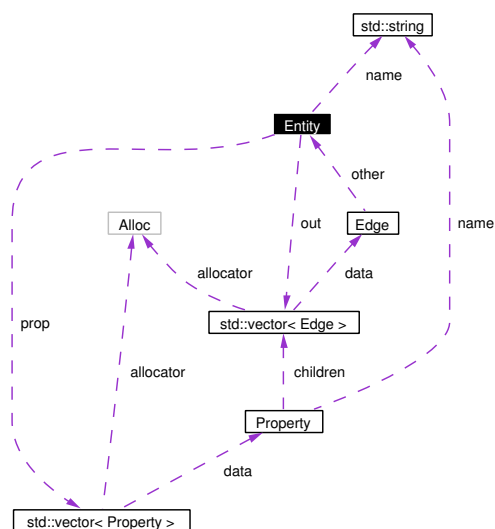
Definition of Entity class and its functions.

```
#include <Graph.H>
```

Inheritance diagram for Entity:



Collaboration diagram for Entity:



### Public Member Functions

- `Entity ()`  
*Constructor.*
- `virtual ~Entity ()`  
*Destructor.*

- void `reset ()`  
*set entity to default values*
- void `set_name (std::string name)`
- void `set_flag (int flag)`
- void `add_outgoing_edge (int index, Edge &edge)`  
*add an edge to outgoing edge*
- void `add_property (Property &p)`
- `std::string get_name () const`  
*get name of entity*
- int `get_flag () const`  
*get the flag of entity*
- void `get_incoming_edges (std::vector< Edge > *in)`
- const `std::vector< Edge > * get_outgoing_edges () const`  
*get outgoing edges*
- const `std::vector< Property > get_property () const`  
*get properties*
- virtual float `get_relevance () const`
- virtual void `set_relevance (float)`
- virtual void `print_entity () const`
- `std::vector< Edge > * get_outgoing_edges_mutable ()`

## Protected Attributes

- `std::string name`
- int `flag`
- `std::vector< Edge > out [SIZE]`
- `std::vector< Property > prop`

### 10.83.1 Detailed Description

Definition of Entity class and its functions.

The vertices or nodes in the ontology graph are of this type. Each Entity carries a flag showing whether it is real or abstract. It also contains the list of other vertices that it is connected to through the various relations. It has a set of incoming and outgoing edges. Typically, each array index is for relations of a particular type: 0-is a, 1-includes, 2-part of, 3-contains, 4-similar, 5-related. Similarly, it contains a list of properties.

Definition at line 106 of file Graph.H.

## 10.83.2 Constructor & Destructor Documentation

### 10.83.2.1 Entity::Entity () [inline]

Constructor.

Definition at line 287 of file Graph.H.

Referenced by Entity\_TWM::Entity\_TWM().

### 10.83.2.2 Entity::~~Entity () [inline, virtual]

Destructor.

Definition at line 295 of file Graph.H.

## 10.83.3 Member Function Documentation

### 10.83.3.1 void Entity::add\_outgoing\_edge (int *index*, [Edge](#) & *edge*) [inline]

add an edge to outgoing edge

**Parameters:**

*index* enter type of relation ie 0-is a, 1-includes, 2-part of, 3-contains, 4-similar, 5-related.

*edge* enter the edge to be added

Definition at line 350 of file Graph.H.

### 10.83.3.2 void Entity::add\_property ([Property](#) & *p*) [inline]

add a property to properties list

**Parameters:**

*p* enter a reference to the property to be added

Definition at line 307 of file Graph.H.

### 10.83.3.3 int Entity::get\_flag () const [inline]

get the flag of entity

Definition at line 315 of file Graph.H.

Referenced by Entity\_TWM::Entity\_TWM(), and Map::print\_map().



**10.83.3.4** `void Entity::get_incoming_edges (std::vector< Edge > * in)`  
[inline]

get incoming edges

**Parameters:**

*in* enter a dummy to which the incoming edges will be assigned

Definition at line 358 of file Graph.H.

Referenced by TWM::add\_compute\_relevance().

**10.83.3.5** `std::string Entity::get_name () const` [inline]

get name of entity

Definition at line 311 of file Graph.H.

Referenced by TWM::add\_compute\_relevance(), TWM::check\_relevance(), Entity\_TWM::Entity\_TWM(), TWM::insert(), Map::print\_map(), and TWM::propagate\_relevance().

**10.83.3.6** `const std::vector< Edge > * Entity::get_outgoing_edges () const`  
[inline]

get outgoing edges

Definition at line 319 of file Graph.H.

Referenced by Entity\_TWM::Entity\_TWM().

**10.83.3.7** `const std::vector< Property > Entity::get_property () const`  
[inline]

get properties

Definition at line 323 of file Graph.H.

Referenced by Entity\_TWM::Entity\_TWM().

**10.83.3.8** `void Entity::reset ()` [inline]

set entity to default values

Definition at line 335 of file Graph.H.

**10.83.3.9 void Entity::set\_flag (int *flag*) [inline]**

set the flag of entity

**Parameters:**

*flag* enter flag of entity

Definition at line 303 of file Graph.H.

**10.83.3.10 void Entity::set\_name (std::string *name*) [inline]**

set name of entity

**Parameters:**

*name* enter name of entity

Definition at line 299 of file Graph.H.

The documentation for this class was generated from the following files:

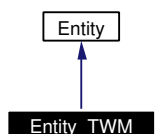
- [Graph.H](#)
- [Graph.C](#)

## 10.84 Entity\_TWM Class Reference

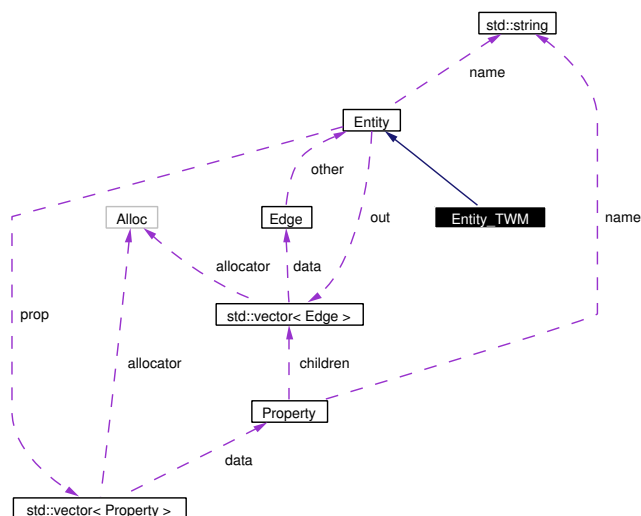
Definition of Entity\_TWM class and its functions.

```
#include <Graph.H>
```

Inheritance diagram for Entity\_TWM:



Collaboration diagram for Entity\_TWM:



### Public Member Functions

- [Entity\\_TWM](#) (const [Entity](#) &e)

*Constructor.*

- [Entity\\_TWM](#) ()
- [~Entity\\_TWM](#) ()

*Destructor.*

- void [set\\_relevance](#) (float r)
- float [get\\_relevance](#) () const  
*get relevance of entity\_twm*
- [std::vector< Edge > \\* get\\_outgoing\\_edges](#) ()  
*get a mutable access to outgoing edges*
- void [set\\_outgoing\\_edges](#) (std::vector< [Edge](#) > out\_edges[SIZE])  
*set outgoing edges*
- void [print\\_entity](#) () const  
*Print entity\_twm.*

### 10.84.1 Detailed Description

Definition of Entity\_TWM class and its functions.

This class inherits from [Entity](#) class. It contains an estimate of relevance associated with the [Entity](#). Nodes in task graph are of this type

Definition at line 175 of file Graph.H.

### 10.84.2 Constructor & Destructor Documentation

#### 10.84.2.1 Entity\_TWM::Entity\_TWM (const [Entity](#) & e) [inline]

Constructor.

This constructor takes a reference to an [Entity](#) object as argument and creates the corresponding Entity\_TWM object. It copies all member values of entity into the new entity\_twm object

Definition at line 401 of file Graph.H.

References [Entity::get\\_flag\(\)](#), [Entity::get\\_name\(\)](#), [Entity::get\\_outgoing\\_edges\(\)](#), and [Entity::get\\_property\(\)](#).

#### 10.84.2.2 Entity\_TWM::Entity\_TWM () [inline]

This constructor initialises Entity\_TWM object with default values

Definition at line 378 of file Graph.H.

References [Entity::Entity\(\)](#).

### 10.84.2.3 Entity\_TWM::~~Entity\_TWM () [inline]

Destructor.

Definition at line 385 of file Graph.H.

## 10.84.3 Member Function Documentation

### 10.84.3.1 std::vector< Edge > \* Entity\_TWM::get\_outgoing\_edges () [inline]

get a mutable access to outgoing edges

Definition at line 397 of file Graph.H.

### 10.84.3.2 float Entity\_TWM::get\_relevance () const [inline, virtual]

get relevance of entity\_twm

Reimplemented from [Entity](#).

Definition at line 393 of file Graph.H.

Referenced by TWM::add\_compute\_relevance().

### 10.84.3.3 void Entity\_TWM::print\_entity () const [virtual]

Print entity\_twm.

Reimplemented from [Entity](#).

Definition at line 62 of file Graph.C.

### 10.84.3.4 void Entity\_TWM::set\_outgoing\_edges (std::vector< Edge > out\_edges[SIZE]) [inline]

set outgoing edges

Definition at line 419 of file Graph.H.

### 10.84.3.5 void Entity\_TWM::set\_relevance (float r) [inline, virtual]

set relevance of entity\_twm

#### Parameters:

*r* enter the relevance of this entity

Reimplemented from [Entity](#).

Definition at line 389 of file Graph.H.

Referenced by TWM::add\_compute\_relevance().

The documentation for this class was generated from the following files:

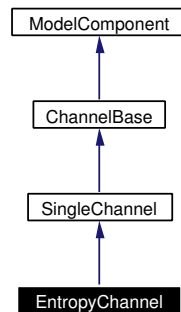
- [Graph.H](#)
- [Graph.C](#)

## 10.85 EntropyChannel Class Reference

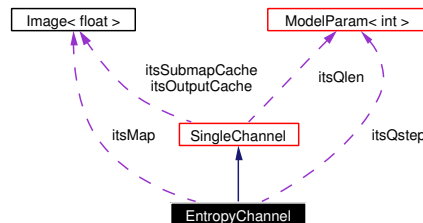
An entropy channel.

```
#include <Channels.H>
```

Inheritance diagram for EntropyChannel:



Collaboration diagram for EntropyChannel:



### Public Member Functions

- [EntropyChannel](#) ([ModelManager](#) &mgr)  
*Construct with standard params.*
- virtual [~EntropyChannel](#) ()  
*Destructor.*
- virtual bool [outputAvailable](#) () const  
*overload*

## Protected Member Functions

- virtual void `doInput` (const `Image< PixRGB< float > > *`, const `Image< float > *bwimg`, const double `t`, const `Image< byte > &clipMask`)  
*special input function*
- virtual `Image< float > getOutput ()`  
*special output function*

## Protected Attributes

- `ModelParam< int > itsQstep`  
*greyscale quantization step*
- `Image< float > itsMap`  
*our entropy map*

### 10.85.1 Detailed Description

An entropy channel.

CAUTION: this channel will throw a fatal exception if your `LevelSpec` contains more than one level or if that level is different from the `maplevel` or if the `deltas` are non-zero.

Definition at line 475 of file `Channels.H`.

### 10.85.2 Constructor & Destructor Documentation

#### 10.85.2.1 `EntropyChannel::EntropyChannel (ModelManager & mgr)`

Construct with standard params.

Definition at line 2496 of file `Channels.C`.

References `ENTROPY`.

#### 10.85.2.2 `EntropyChannel::~EntropyChannel ()` [virtual]

Destructor.

Definition at line 2504 of file `Channels.C`.



### 10.85.3 Member Function Documentation

**10.85.3.1** `void EntropyChannel::doInput (const Image< PixRGB< float > > *, const Image< float > * bwimg, const double t, const Image< byte > & clipMask)` [protected, virtual]

special input function

Reimplemented from [SingleChannel](#).

Definition at line 2512 of file Channels.C.

References ASSERT, [Image](#)< T >::begin(), [Image](#)< float >::beginw(), byte, LevelSpec::delMax(), LevelSpec::delMin(), [Image](#)< T >::getHeight(), ModelParam< int >::getVal(), ModelParam< LevelSpec >::getVal(), [Image](#)< T >::getWidth(), highThresh(), itsMap, itsQstep, LevelSpec::levMax(), LevelSpec::levMin(), LevelSpec::mapLevel(), [Image](#)< float >::resize(), and uint.

**10.85.3.2** `Image< float > EntropyChannel::getOutput ()` [protected, virtual]

special output function

Reimplemented from [SingleChannel](#).

Definition at line 2572 of file Channels.C.

References itsMap.

**10.85.3.3** `bool EntropyChannel::outputAvailable () const` [virtual]

overload

Reimplemented from [SingleChannel](#).

Definition at line 2508 of file Channels.C.

References [Image](#)< float >::initialized(), and itsMap.

### 10.85.4 Member Data Documentation

**10.85.4.1** `Image<float> EntropyChannel::itsMap` [protected]

our entropy map

Definition at line 498 of file Channels.H.

Referenced by doInput(), getOutput(), and outputAvailable().

**10.85.4.2**   [ModelParam<int>](#)   [EntropyChannel::itsQstep](#)   [protected]

greyscale quantization step

Definition at line 488 of file Channels.H.

Referenced by doInput().

The documentation for this class was generated from the following files:

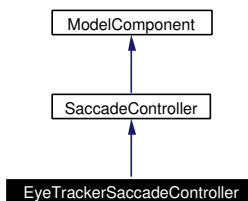
- [Channels.H](#)
- [Channels.C](#)

## 10.86 EyeTrackerSaccadeController Class Reference

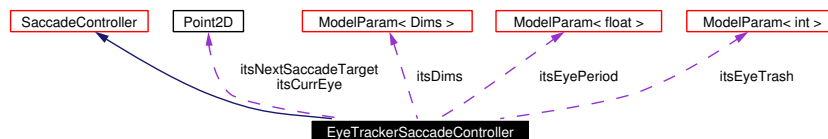
EyeTracker saccade controller.

```
#include <SaccadeControllers.H>
```

Inheritance diagram for EyeTrackerSaccadeController:



Collaboration diagram for EyeTrackerSaccadeController:



### Public Member Functions

- [EyeTrackerSaccadeController](#) ([ModelManager](#) &mgr)  
*Constructor.*
- virtual [~EyeTrackerSaccadeController](#) ()  
*Destructor.*
- virtual void [exportOptions](#) (const int whatToExport, const bool recurse=true)  
*Export some of our [ModelParam](#) parameters as command-line-options.*
- virtual void [evolve](#) (const double t)  
*evolve one time step (called, e.g., by [Brain::evolve\(\)](#))*
- void [getExtraEyeData](#) ([Point2D](#) &nextSacTarget, double &nextSacAmpl, double &nextFixDuration)  
*Get all the extra eye movement data.*

- double [getEyePeriod](#) () const  
*get the eye movement sampling period, in seconds*

## Protected Member Functions

- virtual void [start1](#) ()  
*Get started.*
- virtual void [stop1](#) ()  
*get stopped*
- virtual void [computeWhenNewPercept](#) ()  
*This method is called each time a new percept has arrived.*
- virtual void [computeWhenResetPos](#) ()  
*This method is called each time a reset is made.*
- virtual [Point2D](#) [computeWhenNewDecision](#) (const double t)  
*This method is called each time a new decision is requested.*

## Protected Attributes

- [ModelParam](#)< [Dims](#) > [itsDims](#)  
*Image size so that we stay inside it.*
- [ModelParam](#)< [std::string](#) > [itsEyeFname](#)  
*Name of a calibrated eye movement file.*
- [ModelParam](#)< [float](#) > [itsEyePeriod](#)  
*Period (in s) at which eye movement samples should be read in.*
- [ModelParam](#)< [int](#) > [itsEyeTrash](#)  
*Number of initial eye movement samples to trash.*

### 10.86.1 Detailed Description

EyeTracker saccade controller.

EyeTrackerSaccadeController is driven by a text file that describes human eye position over time, as recorded from an eye-tracker apparatus. The text file format should be: fixed sampling frequency (default 240Hz), one line per sample:

```
ex ey status dx dy sx sy ampl dur
```

where (ex, ey) are the coordinates of eye fixation (point of regard); status is 0 during fixation/smooth pursuit, 1 during saccades, 2 during blinks, 3 during blinks-during-saccades; (dx, dy) coordinates of saccade target location (valid only at start of a saccade, and the following measures are too); (sx, sy) stdev of those target coords over up to 0.25s after end of saccade (might be shorted if next saccade starts sooner then 0.25s after current one); ampl is the amplitude of the saccade, in pixels; dur is the duration of the saccade (in eye-tracker samples).

Definition at line 663 of file SaccadeControllers.H.

## 10.86.2 Constructor & Destructor Documentation

### 10.86.2.1 EyeTrackerSaccadeController::EyeTrackerSaccadeController ([ModelManager](#) & mgr)

Constructor.

Definition at line 1675 of file SaccadeControllers.C.

### 10.86.2.2 EyeTrackerSaccadeController::~~EyeTrackerSaccadeController () [virtual]

Destructor.

Definition at line 1689 of file SaccadeControllers.C.

## 10.86.3 Member Function Documentation

### 10.86.3.1 [Point2D](#) EyeTrackerSaccadeController::computeWhenNewDecision (const double t) [protected, virtual]

This method is called each time a new decision is requested.

Implements [SaccadeController](#).

Definition at line 1843 of file SaccadeControllers.C.

### 10.86.3.2 `void EyeTrackerSaccadeController::computeWhenNewPercept ()` [protected, virtual]

This method is called each time a new percept has arrived.

Implements [SaccadeController](#).

Definition at line 1832 of file `SaccadeControllers.C`.

### 10.86.3.3 `void EyeTrackerSaccadeController::computeWhenResetPos ()` [protected, virtual]

This method is called each time a reset is made.

Implements [SaccadeController](#).

Definition at line 1839 of file `SaccadeControllers.C`.

### 10.86.3.4 `void EyeTrackerSaccadeController::evolve (const double t)` [virtual]

evolve one time step (called, e.g., by [Brain::evolve\(\)](#))

Implements [SaccadeController](#).

Definition at line 1749 of file `SaccadeControllers.C`.

References [SaccadeController::beginBlink\(\)](#), [SaccadeController::beginSaccade\(\)](#), [SaccadeController::endBlink\(\)](#), [SaccadeController::endSaccade\(\)](#), [SharedPtr< T >::get\(\)](#), [Brain::getRET\(\)](#), [ModelParam< int >::getVal\(\)](#), [ModelParam< float >::getVal\(\)](#), [ModelParam< Dims >::getVal\(\)](#), [Dims::h\(\)](#), [Point2D::i](#), [SaccadeController::isInBlink\(\)](#), [SaccadeController::isInSaccade\(\)](#), [itsEyePeriod](#), [itsEyeTrash](#), [Point2D::j](#), [SaccadeController::killDecisions\(\)](#), and [Dims::w\(\)](#).

### 10.86.3.5 `void EyeTrackerSaccadeController::exportOptions (const int whatToExport, const bool recurse = true)` [virtual]

Export some of our [ModelParam](#) parameters as command-line-options.

See the base function in [ModelComponent.H](#) for info.

Reimplemented from [ModelComponent](#).

Definition at line 1693 of file `SaccadeControllers.C`.

References [ModelComponent::exportOptions\(\)](#), [itsEyeFname](#), [itsEyePeriod](#), [itsEyeTrash](#), [OPTEXP\\_CORE](#), and [ModelManager::requestOption\(\)](#).

**10.86.3.6 void EyeTrackerSaccadeController::getExtraEyeData ([Point2D](#) & *nextSacTarget*, double & *nextSacAmpl*, double & *nextFixDuration*)**

Get all the extra eye movement data.

Get all the extra eye movement data from the data file.

**Parameters:**

***nextSacTarget*** coordinates of next saccade target. This is updated each time a saccade starts; so typically, the caller should get this info as soon as the saccade controller has entered a saccade.

***nextSacAmpl*** amplitude of the saccade, in pixels.

***nextFixDuration*** duration of the next fixation period, in s.

Definition at line 1736 of file SaccadeControllers.C.

**10.86.3.7 double EyeTrackerSaccadeController::getEyePeriod () const**

get the eye movement sampling period, in seconds

Definition at line 1850 of file SaccadeControllers.C.

References [ModelParam< float >::getVal\(\)](#), and [itsEyePeriod](#).

**10.86.3.8 void EyeTrackerSaccadeController::start1 () [protected, virtual]**

Get started.

Reimplemented from [ModelComponent](#).

Definition at line 1709 of file SaccadeControllers.C.

References [ModelParam< int >::getVal\(\)](#), [itsEyeFname](#), and [itsEyeTrash](#).

**10.86.3.9 void EyeTrackerSaccadeController::stop1 () [protected, virtual]**

get stopped

Reimplemented from [ModelComponent](#).

Definition at line 1729 of file SaccadeControllers.C.

## 10.86.4 Member Data Documentation

### 10.86.4.1 [ModelParam<Dims> EyeTrackerSaccadeController::itsDims](#) [protected]

[Image](#) size so that we stay inside it.

Definition at line 695 of file SaccadeControllers.H.

### 10.86.4.2 [ModelParam<std::string> EyeTrackerSaccadeController::itsEyeFname](#) [protected]

Name of a calibrated eye movement file.

Definition at line 698 of file SaccadeControllers.H.

Referenced by `exportOptions()`, and `start1()`.

### 10.86.4.3 [ModelParam<float> EyeTrackerSaccadeController::itsEyePeriod](#) [protected]

Period (in s) at which eye movement samples should be read in.

Definition at line 701 of file SaccadeControllers.H.

Referenced by `evolve()`, `exportOptions()`, and `getEyePeriod()`.

### 10.86.4.4 [ModelParam<int> EyeTrackerSaccadeController::itsEyeTrash](#) [protected]

Number of initial eye movement samples to trash.

Definition at line 704 of file SaccadeControllers.H.

Referenced by `evolve()`, `exportOptions()`, and `start1()`.

The documentation for this class was generated from the following files:

- [SaccadeControllers.H](#)
- [SaccadeControllers.C](#)

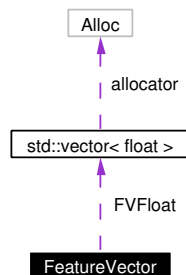


## 10.87 FeatureVector Class Reference

Feature Vector.

```
#include <Keypoint.H>
```

Collaboration diagram for FeatureVector:



### Public Member Functions

- [FeatureVector](#) (int length)  
*Constructor.*
- [~FeatureVector](#) ()  
*Destructor.*
- float [getValue](#) (int index) const  
*get a value from the feature vector*
- void [addValue](#) (float x, float y, float orientation, float value)  
*add value*
- void [normalize](#) ()  
*normalize Vector*
- void [threshold](#) (float limit)  
*threshold Vector and renormalize*

### 10.87.1 Detailed Description

Feature Vector.

Stores temporary keypoint features The final feature vector is formed with 'unsigned char'. This one is just float

Definition at line 146 of file Keypoint.H.

## 10.87.2 Constructor & Destructor Documentation

### 10.87.2.1 **FeatureVector::FeatureVector (int *length*)** [inline]

Constructor.

Definition at line 150 of file Keypoint.H.

### 10.87.2.2 **FeatureVector::~FeatureVector ()** [inline]

Destructor.

Definition at line 156 of file Keypoint.H.

## 10.87.3 Member Function Documentation

### 10.87.3.1 **void FeatureVector::addValue (float *x*, float *y*, float *orientation*, float *value*)**

add value

linearly add a value. Each couple (x,y) will influence 4 positions in the vector

Definition at line 366 of file Keypoint.C.

### 10.87.3.2 **float FeatureVector::getValue (int *index*) const**

get a value from the feature vector

Definition at line 455 of file Keypoint.C.

References uint.

### 10.87.3.3 **void FeatureVector::normalize ()**

normalize Vector

Definition at line 423 of file Keypoint.C.

References sqrt(), and uint.

Referenced by threshold().

#### 10.87.3.4 void FeatureVector::threshold (float *limit*)

threshold Vector and renormalize

Definition at line 439 of file Keypoint.C.

References [normalize\(\)](#), and [uint](#).

The documentation for this class was generated from the following files:

- [Keypoint.H](#)
- [Keypoint.C](#)

## 10.88 FeedForwardNetwork Class Reference

A Feed Forward Network.

```
#include <FFN.H>
```

### Public Member Functions

- [FeedForwardNetwork \(\)](#)  
*Constructor.*
- [~FeedForwardNetwork \(\)](#)  
*Destructor.*
- void [init](#) (char \*wh\_file, char \*wo\_file, int inunits, int hidunits, int outunits, double learnrate, double momentum)  
*Initialize a two-weight-layer network with saved weights.*
- void [init](#) (int inunits, int hidunits, int outunits, double learnrate, double momentum)  
*Initialize a two-weight-layer network with small random weights.*
- void [init3L](#) (char \*wh1\_file, char \*wh2\_file, char \*wo\_file, int inunits, int hid1units, int hid2units, int outunits, double learnrate, double momentum)  
*Initialize a three-weight-layer network with saved weights.*
- void [init3L](#) (int inunits, int hid1units, int hid2units, int outunits, double learnrate, double momentum)  
*Initialize a two-weight-layer network with small random weights.*
- void [run](#) (double input[ ])   
*Run a two-weight-layer network with passed in input.*
- void [run3L](#) (double input[ ])   
*Run a three-weight-layer network with passed in input.*
- void [backprop](#) (double \*target)  
*backpropagate the error in the two-weight-layer network*
- void [backprop3L](#) (double \*target)  
*backpropagate the error in the three-weight-layer network*
- void [write](#) (char \*wh\_file, char \*wo\_file)

*Write the two-weight-layer network weights to a specified file.*

- void [write3L](#) (char \*wh1\_file, char \*wh2\_file, char \*wo\_file)

*Write the three-weight-layer network weights to a specified file.*

## Public Attributes

- double \* **out**

### 10.88.1 Detailed Description

A Feed Forward Network.

Definition at line 50 of file FFN.H.

### 10.88.2 Constructor & Destructor Documentation

#### 10.88.2.1 FeedForwardNetwork::FeedForwardNetwork ()

Constructor.

This implements a simple feedforward network with backpropagation. Supports 2 or 3-weight-layer networks through separate function calls NOTE: please allow the class to create the initial weight files (use the init function that does not ask for weight files). Also user should add an extra input for bias (set it to 1.0)

Definition at line 56 of file FFN.C.

#### 10.88.2.2 FeedForwardNetwork::~FeedForwardNetwork ()

Destructor.

Definition at line 572 of file FFN.C.

### 10.88.3 Member Function Documentation

#### 10.88.3.1 void FeedForwardNetwork::backprop (double \* target)

backpropagate the error in the two-weight-layer network

Definition at line 443 of file FFN.C.

**10.88.3.2 void FeedForwardNetwork::backprop3L (double \* target)**

backpropagate the error in the three-weight-layer network

Definition at line 476 of file FFN.C.

**10.88.3.3 void FeedForwardNetwork::init (int inunits, int hidunits, int outunits, double learnrate, double momentum)**

Initialize a two-weight-layer network with small random weights.

Definition at line 70 of file FFN.C.

References init().

**10.88.3.4 void FeedForwardNetwork::init (char \* wh\_file, char \* wo\_file, int inunits, int hidunits, int outunits, double learnrate, double momentum)**

Initialize a two-weight-layer network with saved weights.

Definition at line 79 of file FFN.C.

Referenced by init().

**10.88.3.5 void FeedForwardNetwork::init3L (int inunits, int hid1units, int hid2units, int outunits, double learnrate, double momentum)**

Initialize a two-weight-layer network with small random weights.

Definition at line 209 of file FFN.C.

References init3L().

**10.88.3.6 void FeedForwardNetwork::init3L (char \* wh1\_file, char \* wh2\_file, char \* wo\_file, int inunits, int hid1units, int hid2units, int outunits, double learnrate, double momentum)**

Initialize a three-weight-layer network with saved weights.

Definition at line 219 of file FFN.C.

Referenced by init3L().

**10.88.3.7 void FeedForwardNetwork::run (double input[ ])**

Run a two-weight-layer network with passed in input.

Definition at line 376 of file FFN.C.

#### 10.88.3.8 void FeedForwardNetwork::run3L (double *input*[ ])

Run a three-weight-layer network with passed in input.

Definition at line 407 of file FFN.C.

#### 10.88.3.9 void FeedForwardNetwork::write (char \* *wh\_file*, char \* *wo\_file*)

Write the two-weight-layer network weights to a specified file.

Definition at line 519 of file FFN.C.

#### 10.88.3.10 void FeedForwardNetwork::write3L (char \* *wh1\_file*, char \* *wh2\_file*, char \* *wo\_file*)

Write the three-weight-layer network weights to a specified file.

Definition at line 542 of file FFN.C.

The documentation for this class was generated from the following files:

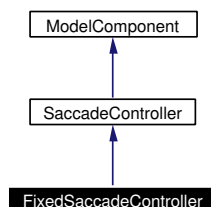
- [FFN.H](#)
- [FFN.C](#)

## 10.89 FixedSaccadeController Class Reference

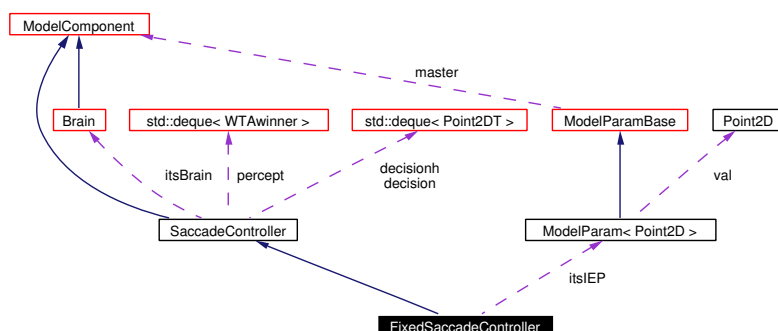
Fixed saccade controller.

```
#include <SaccadeControllers.H>
```

Inheritance diagram for FixedSaccadeController:



Collaboration diagram for FixedSaccadeController:



### Public Member Functions

- [FixedSaccadeController](#) ([ModelManager](#) &mgr)  
*Constructor.*
- virtual [~FixedSaccadeController](#) ()  
*Destructor.*
- virtual void [exportOptions](#) (const int whatToExport, const bool recurse=true)  
*Export some of our [ModelParam](#) parameters as command-line-options.*
- virtual void [evolve](#) (const double t)



*evolve one time step (called, e.g., by [Brain::evolve\(\)](#))*

- void [saccade](#) (const bool insac)  
*externally decide whether we are in a saccade*
- void [blink](#) (const bool inbli)  
*externally decide whether we are in blink*

## Protected Member Functions

- virtual void [computeWhenNewPercept](#) ()  
*This method is called each time a new percept has arrived.*
- virtual void [computeWhenResetPos](#) ()  
*This method is called each time a reset is made.*
- virtual [Point2D](#) [computeWhenNewDecision](#) (const double t)  
*This method is called each time a new decision is requested.*

## Protected Attributes

- [ModelParam](#)< [Point2D](#) > [itsIEP](#)  
*Initial (and permanent) eye position.*

### 10.89.1 Detailed Description

Fixed saccade controller.

FixedSaccadeController holds fixation at a fixed point, ignoring its percepts. Useful to render the difference between covert and overt attention evident in the model. Also useful for creating foveated displays with fixed viewpoint. Like with all saccade controllers, the fixation point may externally be changed using [resetPos\(\)](#). Also here, the controller may be externally set as in saccade or blinking using [saccade\(\)](#) and [blink\(\)](#).

Definition at line 141 of file SaccadeControllers.H.

## 10.89.2 Constructor & Destructor Documentation

### 10.89.2.1 `FixedSaccadeController::FixedSaccadeController (ModelManager & mgr)`

Constructor.

Definition at line 191 of file SaccadeControllers.C.

### 10.89.2.2 `FixedSaccadeController::~FixedSaccadeController ()` [virtual]

Destructor.

Definition at line 198 of file SaccadeControllers.C.

## 10.89.3 Member Function Documentation

### 10.89.3.1 `void FixedSaccadeController::blink (const bool inbli)`

externally decide whether we are in blink

Definition at line 226 of file SaccadeControllers.C.

References `SaccadeController::beginBlink()`, `SaccadeController::endBlink()`, and `SaccadeController::isInBlink()`.

### 10.89.3.2 `Point2D FixedSaccadeController::computeWhenNewDecision (const double t)` [protected, virtual]

This method is called each time a new decision is requested.

Implements [SaccadeController](#).

Definition at line 241 of file SaccadeControllers.C.

References `ModelParam< Point2D >::getVal()`, and `itsIEP`.

### 10.89.3.3 `void FixedSaccadeController::computeWhenNewPercept ()` [protected, virtual]

This method is called each time a new percept has arrived.

Implements [SaccadeController](#).

Definition at line 233 of file SaccadeControllers.C.

#### 10.89.3.4 void FixedSaccadeController::computeWhenResetPos () [protected, virtual]

This method is called each time a reset is made.

Implements [SaccadeController](#).

Definition at line 237 of file SaccadeControllers.C.

References [SaccadeController::getPreviousPercept\(\)](#), [itsIEP](#), and [ModelParam< Point2D >::setVal\(\)](#).

#### 10.89.3.5 void FixedSaccadeController::evolve (const double t) [virtual]

evolve one time step (called, e.g., by [Brain::evolve\(\)](#))

Implements [SaccadeController](#).

Definition at line 215 of file SaccadeControllers.C.

#### 10.89.3.6 void FixedSaccadeController::exportOptions (const int *whatToExport*, const bool *recurse* = true) [virtual]

Export some of our [ModelParam](#) parameters as command-line-options.

See the base function in [ModelComponent.H](#) for info.

Reimplemented from [ModelComponent](#).

Definition at line 202 of file SaccadeControllers.C.

References [ModelComponent::exportOptions\(\)](#), [itsIEP](#), [OPTEXP\\_CORE](#), and [ModelManager::requestOption\(\)](#).

#### 10.89.3.7 void FixedSaccadeController::saccade (const bool *insac*)

externally decide whether we are in a saccade

Definition at line 219 of file SaccadeControllers.C.

References [SaccadeController::beginSaccade\(\)](#), [SaccadeController::endSaccade\(\)](#), and [SaccadeController::isInSaccade\(\)](#).

## 10.89.4 Member Data Documentation

### 10.89.4.1 [ModelParam<Point2D> FixedSaccadeController::itsIEP](#) [protected]

Initial (and permanent) eye position.

Definition at line 174 of file SaccadeControllers.H.

Referenced by [computeWhenNewDecision\(\)](#), [computeWhenResetPos\(\)](#), and [exportOptions\(\)](#).

The documentation for this class was generated from the following files:

- [SaccadeControllers.H](#)
- [SaccadeControllers.C](#)

## 10.90 FlickerChannel Class Reference

A temporal flicker channel.

```
#include <Channels.H>
```

Inheritance diagram for FlickerChannel: Collaboration diagram for FlickerChannel:

### Public Member Functions

- [FlickerChannel](#) ([ModelManager](#) &mgr)  
*Construct with standard specs.*
- virtual [~FlickerChannel](#) ()  
*Destructor.*
- virtual void [learn](#) (const [Image](#)< [byte](#) > &dmap, double eta)  
*This operation does nothing.*
- virtual [Image](#)< float > [getOutput](#) ()  
*get the output*
- virtual void [reset](#) (const bool recurse=true)  
*Reset FlickerChannel.*

### Protected Member Functions

- virtual void [doInput](#) (const [Image](#)< [PixRGB](#)< float > > \*, const [Image](#)< float > > \*bwimg, const double t, const [Image](#)< [byte](#) > &clipMask)  
*FlickerChannel requires only luminance input.*

### Protected Attributes

- [ModelParam](#)< double > [itsTimeConst](#)
- [Image](#)< float > [itsPrevInput](#)
- double [itsPrevTime](#)
- [Image](#)< float > [itsOutput](#)

### 10.90.1 Detailed Description

A temporal flicker channel.

This channel just keeps a memory trace of the previous frame and then does a basic center-surround processing on the difference between current and memorized input. The time constant determines how the memorized input should be updated when a new input arrives.

Definition at line 276 of file Channels.H.

### 10.90.2 Constructor & Destructor Documentation

#### 10.90.2.1 `FlickerChannel::FlickerChannel (ModelManager & mgr)`

Construct with standard specs.

Definition at line 390 of file Channels.C.

References `FLICKER`, `makeShared()`, and `ModelParam< bool >::setVal()`.

#### 10.90.2.2 `FlickerChannel::~FlickerChannel ()` [virtual]

Destructor.

Definition at line 403 of file Channels.C.

### 10.90.3 Member Function Documentation

#### 10.90.3.1 `void FlickerChannel::doInput (const Image< PixRGB< float > > *, const Image< float > * bwimg, const double t, const Image< byte > & clipMask)` [protected, virtual]

`FlickerChannel` requires only luminance input.

Reimplemented from [SingleChannel](#).

Reimplemented in [FlickerChannelBeo](#).

Definition at line 412 of file Channels.C.

References `absDiff()`, `ASSERT`, `SingleChannel::doInput()`, `Image< float >::freeMem()`, `ModelParam< double >::getVal()`, `Image< float >::initialized()`, and `itsTimeConst`.

**10.90.3.2** [Image< float >](#) **FlickerChannel::getOutput ()** [virtual]

get the output

Reimplemented from [SingleChannel](#).

Definition at line 454 of file Channels.C.

References [SingleChannel::combineSubMaps\(\)](#), [Image< float >::getDims\(\)](#), [ModelParam< MaxNormType >::getVal\(\)](#), [ModelParam< float >::getVal\(\)](#), [ModelParam< bool >::getVal\(\)](#), [ModelParam< LevelSpec >::getVal\(\)](#), [Dims::h\(\)](#), [Image< float >::initialized\(\)](#), [LevelSpec::mapLevel\(\)](#), [maxNormalize\(\)](#), [SingleChannel::outputAvailable\(\)](#), [Dims::w\(\)](#), and [ZEROS](#).

**10.90.3.3** **void FlickerChannel::learn (const [Image< byte >](#) & *dmap*, double *eta*)** [virtual]

This operation does nothing.

That's because FlickerChannel cannot [learn\(\)](#) from a single input.

Definition at line 406 of file Channels.C.

**10.90.3.4** **void FlickerChannel::reset (const bool *recurse* = true)** [virtual]

Reset FlickerChannel.

See the base function in [ModelComponent.H](#) for info.

Reimplemented from [SingleChannel](#).

Definition at line 482 of file Channels.C.

References [Image< float >::freeMem\(\)](#), and [SingleChannel::reset\(\)](#).

**10.90.4 Member Data Documentation****10.90.4.1** [ModelParam<double>](#) **FlickerChannel::itsTimeConst** [protected]

Time constant of the channel, in seconds. This is used to determine how to update our memory of the previous frame as a new frame arrives. If the previous frame is more than a time constant older than the new one, it will be replaced by the new one for the next step of processing; otherwise a mix between both will be used, depending on the difference between old and new times.

Definition at line 309 of file Channels.H.

Referenced by [doInput\(\)](#).

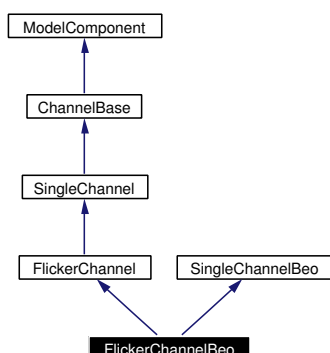
The documentation for this class was generated from the following files:

- [Channels.H](#)
- [Channels.C](#)



Beowulf version of [FlickerChannel](#).

Inheritance diagram for FlickerChannelBeo:



- **FlickerChannelBeo** ([ModelManager](#) &mgr, const [SharedPtr](#)< [Beowulf](#) > &beow)
- virtual bool [outputAvailable](#) () const

Generated on Mon Nov 8 04:53:06 2004 for iLab Neuromorphic Vision Toolkit by Doxygen

## Protected Member Functions

- virtual void [doInput](#) (const [Image](#)< [PixRGB](#)< float > > \*, const [Image](#)< float > \*bwimg, const double t, const [Image](#)< byte > &clipMask)  
*[FlickerChannel](#) requires only luminance input.*
- virtual void [start1](#) ()

### 10.91.1 Detailed Description

[Beowulf](#) version of [FlickerChannel](#).

Definition at line 113 of file ChannelsBeo.H.

### 10.91.2 Member Function Documentation

**10.91.2.1 void [FlickerChannelBeo::doInput](#) (const [Image](#)< [PixRGB](#)< float > > \*, const [Image](#)< float > \*bwimg, const double t, const [Image](#)< byte > &clipMask) [protected, virtual]**

[FlickerChannel](#) requires only luminance input.

Reimplemented from [FlickerChannel](#).

Definition at line 162 of file ChannelsBeo.C.

References [Image](#)< float >::freeMem(), and [SingleChannelBeo::sendInput](#)().

**10.91.2.2 bool [FlickerChannelBeo::outputAvailable](#) () const [virtual]**

Query whether the channel is ready to give yield valid output.

In some cases (e.g. with motion channels), it may take several input frames before the channel is fully initialized.

Reimplemented from [SingleChannel](#).

Definition at line 149 of file ChannelsBeo.C.

References [Image](#)< float >::initialized(), [FlickerChannel::itsOutput](#), [SingleChannel::outputAvailable](#)(), and [SingleChannelBeo::waitForOutput](#)().

**10.91.2.3 void [FlickerChannelBeo::start1](#) () [protected, virtual]**

If you overload this, make sure you call [SingleChannel::start1\(\)](#) at the beginning of your overload

Reimplemented from [SingleChannel](#).

Definition at line 179 of file ChannelsBeo.C.

References [SingleChannelBeo::requestBeowulfNode\(\)](#), and [SingleChannel::start1\(\)](#).

The documentation for this class was generated from the following files:

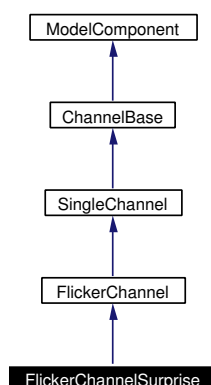
- [ChannelsBeo.H](#)
- [ChannelsBeo.C](#)

## 10.92 FlickerChannelSurprise Class Reference

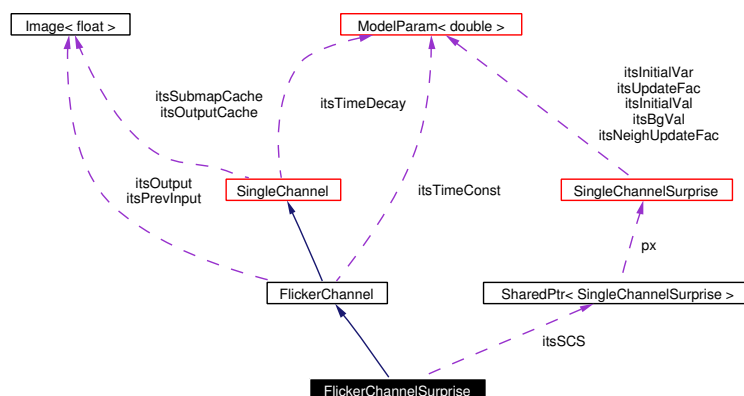
A temporal flicker channel.

```
#include <ChannelsSurprise.H>
```

Inheritance diagram for FlickerChannelSurprise:



Collaboration diagram for FlickrChannelSurprise:



## Public Member Functions

- `FlickerChannelSurprise (ModelManager &mgr)`

*Construct with standard specs.*

- virtual [~FlickerChannelSurprise](#) ()  
*Destructor.*
- virtual [Image](#)< float > [computeSubmap](#) (const [uint](#) index) const  
*[computeSubmap\(\)](#) is special and will return surprise*

## Protected Attributes

- [SharedPtr](#)< [SingleChannelSurprise](#) > [itsSCS](#)  
*our surprise computer*

### 10.92.1 Detailed Description

A temporal flicker channel.

Definition at line 286 of file ChannelsSurprise.H.

### 10.92.2 Constructor & Destructor Documentation

#### 10.92.2.1 FlickerChannelSurprise::FlickerChannelSurprise ([ModelManager](#) & *mgr*)

Construct with standard specs.

Definition at line 261 of file ChannelsSurprise.C.

References [ModelComponent::addSubComponent\(\)](#).

#### 10.92.2.2 FlickerChannelSurprise::~~FlickerChannelSurprise () [virtual]

Destructor.

Definition at line 267 of file ChannelsSurprise.C.

### 10.92.3 Member Function Documentation

#### 10.92.3.1 [Image](#)< float > FlickerChannelSurprise::computeSubmap (const [uint](#) index) const [virtual]

[computeSubmap\(\)](#) is special and will return surprise

Reimplemented from [SingleChannel](#).

Definition at line 270 of file ChannelsSurprise.C.

References `uint`.

#### 10.92.4 Member Data Documentation

##### 10.92.4.1 [SharedPtr<SingleChannelSurprise> FlickerChannelSurprise::its-SCS](#) [protected]

our surprise computer

Definition at line 299 of file ChannelsSurprise.H.

The documentation for this class was generated from the following files:

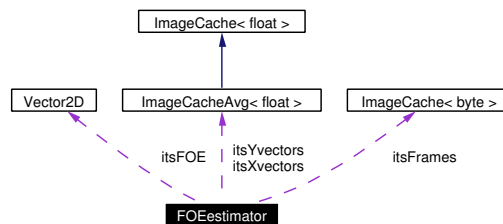
- [ChannelsSurprise.H](#)
- ChannelsSurprise.C

## 10.93 FOEstimator Class Reference

compute the focus of expansion (FOE) from the pixel-based optical flow

```
#include <FOEstimator.H>
```

Collaboration diagram for FOEstimator:



### Public Member Functions

- **FOEstimator** (int numAvg=20, int pyramidLevel=0)  
*constructor*
- **Vector2D** updateFOE (const **Image**< byte > &img, const **Rectangle** region)  
*update with a new frame of the movie sequence*
- **Vector2D** updateFOE (const **Image**< byte > &img)  
*update with a new frame of the movie sequence*
- **Vector2D** getFOE ()  
*returns the last estimate of the FOA*

### 10.93.1 Detailed Description

compute the focus of expansion (FOE) from the pixel-based optical flow

Definition at line 47 of file FOEstimator.H.

### 10.93.2 Constructor & Destructor Documentation

#### 10.93.2.1 FOEstimator::FOEstimator (int numAvg = 20, int pyramidLevel = 0)

constructor

**Parameters:**

- numAvg* the number of frames used for averaging to get a smooth estimate  
*pyramidLevel* the subsampling level at which to compute the optical flow

Definition at line 45 of file FOEstimator.C.

### 10.93.3 Member Function Documentation

#### 10.93.3.1 [Vector2D](#) FOEstimator::getFOE ()

returns the last estimate of the FOA

Definition at line 158 of file FOEstimator.C.

#### 10.93.3.2 [Vector2D](#) FOEstimator::updateFOE (const [Image](#)< [byte](#) > & *img*)

update with a new frame of the movie sequence

**Returns:**

if the cache contains less than three frames, an invalid [Vector2D](#) is returned, otherwise the estimated location of the FOE is returned

Definition at line 61 of file FOEstimator.C.

References [ImageCache](#)< [byte](#) >::back(), [Image](#)< [T](#) >::begin(), [buildPyrGaussian](#)(), [Image](#)< [byte](#) >::getHeight(), [Image](#)< [byte](#) >::getWidth(), [ImageCacheAvg](#)< [float](#) >::mean(), [ImageCache](#)< [float](#) >::push\_back(), [ImageCache](#)< [byte](#) >::push\_back(), [Vector2D](#)::reset(), [ImageCache](#)< [byte](#) >::size(), and [ZEROS](#).

#### 10.93.3.3 [Vector2D](#) FOEstimator::updateFOE (const [Image](#)< [byte](#) > & *img*, const [Rectangle](#) *region*)

update with a new frame of the movie sequence

**Returns:**

if the cache contains less than three frames, an invalid [Vector2D](#) is returned, otherwise the estimated location of the FOE is returned

Definition at line 53 of file FOEstimator.C.

References [crop](#)(), [Rectangle](#)::left(), and [Rectangle](#)::top().

The documentation for this class was generated from the following files:

- [FOEstimator.H](#)
- [FOEstimator.C](#)

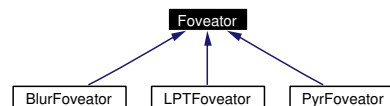


## 10.94 Foveator Class Reference

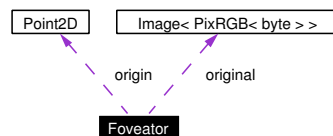
An abstract class for space-variant image processing.

```
#include <Foveator.H>
```

Inheritance diagram for Foveator:



Collaboration diagram for Foveator:



### Public Member Functions

- [Foveator](#) (const [Image< PixRGB< byte > >](#) &img)  
*Constructor.*
- virtual [~Foveator](#) ()  
*Destructor.*
- [Point2D](#) [getOrigin](#) (void) const  
*Returns the current center of foveation.*
- int [getWidth](#) (void) const  
*Returns width of image being foveated.*
- int [getHeight](#) (void) const  
*Returns height of image being foveated.*
- void [changeImage](#) (const [Image< PixRGB< byte > >](#) &img)  
*Change the image to be foveated.*
- virtual bool [setOrigin](#) ([Point2D](#) pt)

*Set origin point (center of foveation).*

- virtual bool [setOrigin](#) (int x, int y)  
*Set origin point (center of foveation).*
- virtual [Image](#)< [PixRGB](#)< [byte](#) > > [foveate](#) (void)=0  
*Foveation function, returns foveated image.*

## Protected Attributes

- [Image](#)< [PixRGB](#)< [byte](#) > > [original](#)  
*image to be foveated*
- int [width](#)  
*width of original image*
- int [height](#)  
*height of original image*
- [Point2D](#) [origin](#)  
*center of foveation*

### 10.94.1 Detailed Description

An abstract class for space-variant image processing.

Space-variant image processing concentrates image quality and resolution in areas of interest. Human vision has high resolution at the center, but it decreases as we move further out into the periphery. Through the process of foveation, we attempt to simulate this type of vision.

Classes used for foveation should inherit this as a base. The foveation operation is performed through pure virtual function [foveate\(\)](#). Different foveation techniques and algorithms are implemented in [BlurFoveator](#), [PyrFoveator](#), and [LPTFoveator](#).

Definition at line 24 of file Foveator.H.

### 10.94.2 Constructor & Destructor Documentation

#### 10.94.2.1 Foveator::Foveator (const [Image](#)< [PixRGB](#)< [byte](#) > > & img)

Constructor.

Create a Foveator object to operate on image argument

**Parameters:**

*img* image to be foveated

Definition at line 9 of file Foveator.C.

References height, and width.

**10.94.2.2 Foveator::~~Foveator () [virtual]**

Destructor.

Definition at line 17 of file Foveator.C.

**10.94.3 Member Function Documentation****10.94.3.1 void Foveator::changeImage (const Image< PixRGB< byte > > & *img*)**

Change the image to be foveated.

Change the image to be foveated to another one of the same size

**Parameters:**

*img* new image to be used, must have same dimensions as original

Definition at line 44 of file Foveator.C.

References height, original, and width.

**10.94.3.2 virtual Image< PixRGB< byte> > Foveator::foveate (void) [pure virtual]**

Foveation function, returns foveated image.

Implemented in [BlurFoveator](#), [LPTFoveator](#), and [PyrFoveator](#).

**10.94.3.3 int Foveator::getHeight (void) const**

Returns height of image being foveated.

Definition at line 36 of file Foveator.C.

References height.

**10.94.3.4 [Point2D](#) Foveator::getOrigin (void) const**

Returns the current center of foveation.

Definition at line 24 of file Foveator.C.

References [origin](#).

**10.94.3.5 int Foveator::getWidth (void) const**

Returns width of image being foveated.

Definition at line 30 of file Foveator.C.

References [width](#).

**10.94.3.6 bool Foveator::setOrigin (int x, int y) [virtual]**

Set origin point (center of foveation).

Set origin point to two int arguments, returns true if successful

**Parameters:**

*x* horizontal coordinate of origin

*y* vertical coordinate of origin

Reimplemented in [LPTFoveator](#).

Definition at line 75 of file Foveator.C.

References [Image< PixRGB< byte > >::coordsOk\(\)](#), [Point2D::i](#), [Point2D::j](#), [origin](#), and [original](#).

**10.94.3.7 bool Foveator::setOrigin ([Point2D](#) pt) [virtual]**

Set origin point (center of foveation).

Set origin point to [Point2D](#) argument, returns true if successful

**Parameters:**

*pt* center of foveation

Reimplemented in [LPTFoveator](#).

Definition at line 60 of file Foveator.C.

References [Image< PixRGB< byte > >::coordsOk\(\)](#), [origin](#), and [original](#).

Referenced by [LPTFoveator::setOrigin\(\)](#).

## 10.94.4 Member Data Documentation

### 10.94.4.1 `int Foveator::height` [protected]

height of original image

Definition at line 29 of file Foveator.H.

Referenced by `changeImage()`, and `getHeight()`.

### 10.94.4.2 `Point2D Foveator::origin` [protected]

center of foveation

Definition at line 30 of file Foveator.H.

Referenced by `getOrigin()`, and `setOrigin()`.

### 10.94.4.3 `Image< PixRGB< byte > > Foveator::original` [protected]

image to be foveated

Definition at line 27 of file Foveator.H.

Referenced by `changeImage()`, and `setOrigin()`.

### 10.94.4.4 `int Foveator::width` [protected]

width of original image

Definition at line 28 of file Foveator.H.

Referenced by `changeImage()`, and `getWidth()`.

The documentation for this class was generated from the following files:

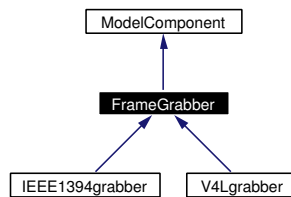
- [Foveator.H](#)
- [Foveator.C](#)

## 10.95 FrameGrabber Class Reference

Base class for a frame grabber device.

```
#include <FrameGrabber.H>
```

Inheritance diagram for FrameGrabber:



Collaboration diagram for FrameGrabber:

### Public Member Functions

- **FrameGrabber** (**ModelManager** &mgr, const **std::string** &descrName="Frame Grabber Driver", const **std::string** &tagName="FrameGrabber")

*Constructor.*

- virtual **~FrameGrabber** ()=0

*Destructor.*

- virtual void **exportOptions** (const int whatToExport, const bool recurse=true)

*Export some of our **ModelParam** parameters as command-line-options.*

- virtual **Image**< **PixRGB**< **byte** > > **grab** ()=0

*Grab a color image.*

- virtual **Image**< **PixRGB**< **byte** > > **grabSingle** ()=0

*Grab a single color image.*

- int **getWidth** () const

*get image width corresponding to selected mode*

- int **getHeight** () const

*get image height corresponding to selected mode*

## Protected Member Functions

- `Image< PixRGB< byte > > grab2rgb` (const `byte *data`) const  
*Use this to convert from a buffer packed according to grabmode to RGB.*

## Protected Attributes

- `ModelParam< std::string > itsDevName`  
*device name of the /dev/ entry for the grabber device*
- `ModelParam< int > itsChannel`  
*input channel to use*
- `ModelParam< int > itsSubChan`  
*input subchannel to use*
- `ModelParam< Dims > itsDims`  
*width of grabbed frames*
- `ModelParam< FrameGrabberMode > itsGrabMode`  
*grab mode that the hardware should use*
- `ModelParam< bool > itsByteSwap`  
*determines whether byte swapping is done during conversion to RGB*
- `ModelParam< float > itsFPS`  
*frames per second*
- `ModelParam< int > itsNbuf`  
*number of frame buffers kept internally*
- `ModelParam< int > itsBrightness`  
*brightness - highly dependent on your driver*
- `ModelParam< int > itsHue`  
*hue - highly dependent on your driver*
- `ModelParam< int > itsSaturation`  
*saturation - highly dependent on your driver*
- `ModelParam< int > itsColour`

*color - highly dependent on your driver*

- [ModelParam](#)< int > [itsContrast](#)

*contrast - highly dependent on your driver*

- [ModelParam](#)< int > [itsWhiteness](#)

*whiteness - highly dependent on your driver*

- [ModelParam](#)< int > [itsExposure](#)

*exposure - highly dependent on your driver*

- [ModelParam](#)< int > [itsSharpness](#)

*sharpness - highly dependent on your driver*

- [ModelParam](#)< int > [itsWhiteBalBU](#)

*white balance (B/U value) - highly dependent on your driver*

- [ModelParam](#)< int > [itsWhiteBalRV](#)

*white balance (R/V value) - highly dependent on your driver*

- [ModelParam](#)< int > [itsGamma](#)

*gamma - highly dependent on your driver*

- [ModelParam](#)< int > [itsShutter](#)

*shutter - highly dependent on your driver*

- [ModelParam](#)< int > [itsGain](#)

*gain - highly dependent on your driver*

### 10.95.1 Detailed Description

Base class for a frame grabber device.

This is an attempt to abstract various framegrabber hardware under a common interface. See the [V4Lgrabber](#) and [IEEE1394grabber](#) classes for examples of derivations from that base.

Definition at line 52 of file FrameGrabber.H.



## 10.95.2 Constructor & Destructor Documentation

### 10.95.2.1 FrameGrabber::FrameGrabber ([ModelManager](#) & *mgr*, const [std::string](#) & *descrName* = "Frame Grabber Driver", const [std::string](#) & *tagName* = "FrameGrabber")

Constructor.

Definition at line 50 of file FrameGrabber.C.

References [FrameGrabberMode](#).

### 10.95.2.2 FrameGrabber::~~FrameGrabber () [pure virtual]

Destructor.

Definition at line 77 of file FrameGrabber.C.

## 10.95.3 Member Function Documentation

### 10.95.3.1 void FrameGrabber::exportOptions (const int *whatToExport*, const bool *recurse* = true) [virtual]

Export some of our [ModelParam](#) parameters as command-line-options.

See the base function in [ModelComponent.H](#) for info.

Reimplemented from [ModelComponent](#).

Definition at line 81 of file FrameGrabber.C.

References [ModelComponent::exportOptions\(\)](#), [itsBrightness](#), [itsByteSwap](#), [itsChannel](#), [itsColour](#), [itsContrast](#), [itsExposure](#), [itsFPS](#), [itsGain](#), [itsGamma](#), [itsGrabMode](#), [itsHue](#), [itsNbuf](#), [itsSaturation](#), [itsSharpness](#), [itsShutter](#), [itsSubChan](#), [itsWhiteBalBU](#), [itsWhiteBalRV](#), [itsWhiteness](#), [OPTEXP\\_CORE](#), and [ModelManager::requestOption\(\)](#).

### 10.95.3.2 int FrameGrabber::getHeight () const

get image height corresponding to selected mode

Definition at line 172 of file FrameGrabber.C.

References [ModelParam< Dims >::getVal\(\)](#), and [Dims::h\(\)](#).

### 10.95.3.3 `int FrameGrabber::getWidth () const`

get image width corresponding to selected mode

Definition at line 168 of file `FrameGrabber.C`.

References `ModelParam< Dims >::getVal()`, and `Dims::w()`.

### 10.95.3.4 `virtual Image< PixRGB<byte> > FrameGrabber::grab () [pure virtual]`

Grab a color image.

Returns grabbed frame. This call will block until a frame is ready and has been grabbed. If you want to do background grabbing with a non-blocking `grab()` that returns whatever the latest grabbed buffer is at the time of call, see the `AsyncFrameGrabber` class.

Implemented in `IEEE1394grabber`, and `V4Lgrabber`.

### 10.95.3.5 `Image< PixRGB<byte> > FrameGrabber::grab2rgb (const byte * data) const [protected]`

Use this to convert from a buffer packed according to grabmode to RGB.

Typically, your overloaded `grab()` will return `grab2rgb(buffer)`

Definition at line 117 of file `FrameGrabber.C`.

References `fromARGB()`, `fromBGR()`, `fromBGRA()`, `fromMono()`, `fromRGB()`, `fromRGB555()`, `fromRGB555bs()`, `fromRGB565()`, `fromRGB565bs()`, `fromYUV410P()`, `fromYUV411()`, `fromYUV411bs()`, `fromYUV411P()`, `fromYUV420P()`, `fromYUV422()`, `fromYUV422bs()`, `fromYUV422P()`, `fromYUV444()`, `fromYUV444bs()`, `fromYUV444P()`, `ModelParam< FrameGrabberMode >::getVal()`, `ModelParam< bool >::getVal()`, `ModelParam< Dims >::getVal()`, `itsByteSwap`, and `itsGrabMode`.

Referenced by `V4Lgrabber::grab()`, `IEEE1394grabber::grab()`, and `V4Lgrabber::grabSingle()`.

### 10.95.3.6 `virtual Image< PixRGB<byte> > FrameGrabber::grabSingle () [pure virtual]`

Grab a single color image.

This function will work better with single-shot grabs (as opposed to video) and, for example, USB webcams

Implemented in `IEEE1394grabber`, and `V4Lgrabber`.

## 10.95.4 Member Data Documentation

### 10.95.4.1 `ModelParam<int> FrameGrabber::itsBrightness` [protected]

brightness - highly dependent on your driver

Definition at line 117 of file FrameGrabber.H.

Referenced by `exportOptions()`.

### 10.95.4.2 `ModelParam<bool> FrameGrabber::itsByteSwap` [protected]

determines whether byte swapping is done during conversion to RGB

Definition at line 108 of file FrameGrabber.H.

Referenced by `exportOptions()`, and `grab2rgb()`.

### 10.95.4.3 `ModelParam<int> FrameGrabber::itsChannel` [protected]

input channel to use

Definition at line 92 of file FrameGrabber.H.

Referenced by `exportOptions()`.

### 10.95.4.4 `ModelParam<int> FrameGrabber::itsColour` [protected]

color - highly dependent on your driver

Definition at line 126 of file FrameGrabber.H.

Referenced by `exportOptions()`.

### 10.95.4.5 `ModelParam<int> FrameGrabber::itsContrast` [protected]

contrast - highly dependent on your driver

Definition at line 129 of file FrameGrabber.H.

Referenced by `exportOptions()`.

### 10.95.4.6 `ModelParam<std::string> FrameGrabber::itsDevName` [protected]

device name of the `/dev/` entry for the grabber device

Definition at line 89 of file FrameGrabber.H.

**10.95.4.7   [ModelParam<Dims>](#) [FrameGrabber::itsDims](#)   [protected]**

width of grabbed frames

Definition at line 98 of file FrameGrabber.H.

**10.95.4.8   [ModelParam<int>](#) [FrameGrabber::itsExposure](#)   [protected]**

exposure - highly dependent on your driver

Definition at line 135 of file FrameGrabber.H.

Referenced by exportOptions().

**10.95.4.9   [ModelParam<float>](#) [FrameGrabber::itsFPS](#)   [protected]**

frames per second

Definition at line 111 of file FrameGrabber.H.

Referenced by exportOptions().

**10.95.4.10   [ModelParam<int>](#) [FrameGrabber::itsGain](#)   [protected]**

gain - highly dependent on your driver

Definition at line 153 of file FrameGrabber.H.

Referenced by exportOptions().

**10.95.4.11   [ModelParam<int>](#) [FrameGrabber::itsGamma](#)   [protected]**

gamma - highly dependent on your driver

Definition at line 147 of file FrameGrabber.H.

Referenced by exportOptions().

**10.95.4.12   [ModelParam<FrameGrabberMode>](#) [FrameGrabber::itsGrab-Mode](#)   [protected]**

grab mode that the hardware should use

Grabbed frames will internally be converted to [Image< PixRGB<byte> >](#) whatever that mode is, but playing with it may influence image quality, maximum achievable framerate, and amounts of CPU doing those conversions to RGB.

Definition at line 105 of file FrameGrabber.H.

Referenced by `exportOptions()`, and `grab2rgb()`.

#### 10.95.4.13 **ModelParam**<int> **FrameGrabber::itsHue** [protected]

hue - highly dependent on your driver

Definition at line 120 of file `FrameGrabber.H`.

Referenced by `exportOptions()`.

#### 10.95.4.14 **ModelParam**<int> **FrameGrabber::itsNbuf** [protected]

number of frame buffers kept internally

Definition at line 114 of file `FrameGrabber.H`.

Referenced by `exportOptions()`.

#### 10.95.4.15 **ModelParam**<int> **FrameGrabber::itsSaturation** [protected]

saturation - highly dependent on your driver

Definition at line 123 of file `FrameGrabber.H`.

Referenced by `exportOptions()`.

#### 10.95.4.16 **ModelParam**<int> **FrameGrabber::itsSharpness** [protected]

sharpness - highly dependent on your driver

Definition at line 138 of file `FrameGrabber.H`.

Referenced by `exportOptions()`.

#### 10.95.4.17 **ModelParam**<int> **FrameGrabber::itsShutter** [protected]

shutter - highly dependent on your driver

Definition at line 150 of file `FrameGrabber.H`.

Referenced by `exportOptions()`.

#### 10.95.4.18 **ModelParam**<int> **FrameGrabber::itsSubChan** [protected]

input subchannel to use

Definition at line 95 of file `FrameGrabber.H`.

Referenced by `exportOptions()`.

**10.95.4.19**   [`ModelParam<int> FrameGrabber::itsWhiteBalBU`](#)  
[protected]

white balance (B/U value) - highly dependent on your driver

Definition at line 141 of file `FrameGrabber.H`.

Referenced by `exportOptions()`.

**10.95.4.20**   [`ModelParam<int> FrameGrabber::itsWhiteBalRV`](#)  
[protected]

white balance (R/V value) - highly dependent on your driver

Definition at line 144 of file `FrameGrabber.H`.

Referenced by `exportOptions()`.

**10.95.4.21**   [`ModelParam<int> FrameGrabber::itsWhiteness`](#) [protected]

whiteness - highly dependent on your driver

Definition at line 132 of file `FrameGrabber.H`.

Referenced by `exportOptions()`.

The documentation for this class was generated from the following files:

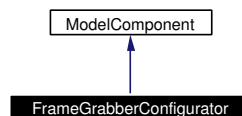
- [FrameGrabber.H](#)
- [FrameGrabber.C](#)

## 10.96 FrameGrabberConfigurator Class Reference

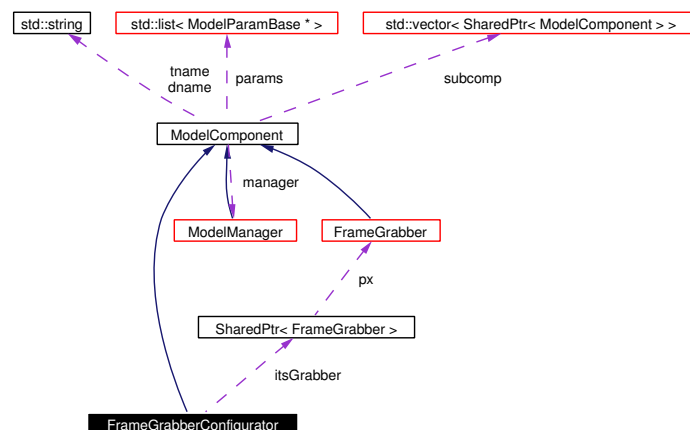
This class is for run-time selection of a frame grabber driver.

```
#include <FrameGrabber.H>
```

Inheritance diagram for FrameGrabberConfigurator:



Collaboration diagram for FrameGrabberConfigurator:



### Public Member Functions

- `FrameGrabberConfigurator` (`ModelManager` &mgr, const `std::string` &descr-Name="Frame Grabber Configurator", const `std::string` &tagName="Frame-GrabberConfigurator")

*Constructor.*

- virtual `~FrameGrabberConfigurator` ()

*destructor*

- virtual void `exportOptions` (const int whatToExport, const bool recurse=true)

*Export some of our `ModelParam` parameters as command-line-options.*

- `SharedPtr< FrameGrabber > getFrameGrabber ()` const

*Get the chosen [FrameGrabber](#).*

## Protected Member Functions

- virtual bool `setModelParamString` (const `std::string` &name, const `std::string` &value, const bool recurse=false, const bool warn=true)

*Intercept people changing our itsGrabberName [ModelParam](#).*

## Protected Attributes

- `ModelParam< std::string > itsGrabberType`

*Type of the grabber.*

### 10.96.1 Detailed Description

This class is for run-time selection of a frame grabber driver.

This will export the `-fg-type=XX` command-line option and will instantiate a [FrameGrabber](#) of the desired type as the option gets assigned a value. As this happens, new options may become available in the command-line. To see them, use `-help` AFTER you have chosen the type of grabber to use. The current grabber may be retrieved using [getFrameGrabber\(\)](#).

Definition at line 168 of file `FrameGrabber.H`.

### 10.96.2 Constructor & Destructor Documentation

- 10.96.2.1** `FrameGrabberConfigurator::FrameGrabberConfigurator`  
 ([ModelManager](#) & mgr, const `std::string` & descrName =  
 "Frame Grabber Configurator", const `std::string` & tagName =  
 "FrameGrabberConfigurator")

Constructor.

Definition at line 178 of file `FrameGrabber.C`.



### 10.96.2.2 FrameGrabberConfigurator::~~FrameGrabberConfigurator () [virtual]

destructor

Definition at line 189 of file FrameGrabber.C.

## 10.96.3 Member Function Documentation

### 10.96.3.1 void FrameGrabberConfigurator::exportOptions (const int *whatToExport*, const bool *recurse* = true) [virtual]

Export some of our [ModelParam](#) parameters as command-line-options.

See the base function in [ModelComponent.H](#) for info. Actually, here we just keep a copy of *whatToExport* and we will call [exportOptions\(\)](#) on our [FrameGrabber](#) once we know its type and instantiate it.

Reimplemented from [ModelComponent](#).

Definition at line 193 of file FrameGrabber.C.

References [ModelComponent::exportOptions\(\)](#), [itsGrabberType](#), [OPTEXP\\_CORE](#), and [ModelManager::requestOption\(\)](#).

### 10.96.3.2 [SharedPtr< \[FrameGrabber\]\(#\) >](#) FrameGrabberConfigurator::get- FrameGrabber () const

Get the chosen [FrameGrabber](#).

You should call this during [start\(\)](#) of the [ModelComponent](#) that needs the [FrameGrabber](#). CAUTION: this may be a [SharedPtr\(NULL\)](#) if no grabber has been chosen.

Definition at line 209 of file FrameGrabber.C.

### 10.96.3.3 bool FrameGrabberConfigurator::setModelParamString (const [std::string](#) & *name*, const [std::string](#) & *value*, const bool *recurse* = false, const bool *warn* = true) [protected, virtual]

Intercept people changing our [itsGrabberName](#) [ModelParam](#).

See [ModelComponent.H](#); as parsing the command-line or reading a config file sets our name, we'll also here instantiate a controller of the proper type (and export its options)

Reimplemented from [ModelComponent](#).

Definition at line 214 of file FrameGrabber.C.

References [ModelComponent::addSubComponent\(\)](#), [SharedPtr< \[FrameGrabber\]\(#\)](#)

>::get(), itsGrabberType, ModelComponent::removeSubComponent(), SharedPtr<FrameGrabber >::reset(), and ModelComponent::setModelParamString().

## 10.96.4 Member Data Documentation

### 10.96.4.1 [ModelParam<std::string> FrameGrabberConfigurator::itsGrabberType](#) [protected]

Type of the grabber.

Definition at line 196 of file FrameGrabber.H.

Referenced by exportOptions(), and setModelParamString().

The documentation for this class was generated from the following files:

- [FrameGrabber.H](#)
- [FrameGrabber.C](#)

## 10.97 FrameRange Class Reference

Specification of a range of frames with given inter-frame delay (in ms).

```
#include <FrameRange.H>
```

### Public Member Functions

- [FrameRange](#) ()  
*Default constructor (first=last=0, delay=0.0F).*
- [FrameRange](#) (const int thefirst, const int thelast, const float thedelay)  
*Initialized constructor.*
- bool [isMultiframe](#) () const  
*Is teh series multiframe (last != first)?*

### Public Attributes

- int [first](#)  
*number of first frame*
- int [last](#)  
*number of last frame*
- float [delay](#)  
*inter-frame delay in ms*

#### 10.97.1 Detailed Description

Specification of a range of frames with given inter-frame delay (in ms).

Definition at line 41 of file FrameRange.H.

#### 10.97.2 Constructor & Destructor Documentation

##### 10.97.2.1 [FrameRange::FrameRange](#) () [inline]

Default constructor (first=last=0, delay=0.0F).

Definition at line 61 of file FrameRange.H.

#### 10.97.2.2 **FrameRange::FrameRange** (const int *thefirst*, const int *thelast*, const float *thedelay*) [inline]

Initialized constructor.

Definition at line 66 of file FrameRange.H.

### 10.97.3 Member Function Documentation

#### 10.97.3.1 **bool FrameRange::isMultiframe ()** const [inline]

Is teh series multiframe (last != first)?

Definition at line 72 of file FrameRange.H.

References first, and last.

Referenced by FrameSeries::computeFileName(), InputFrameSeries::getImageDims(), FrameSeries::isMultiframe(), InputFrameSeries::readGray(), InputFrameSeries::readRGB(), FrameSeries::update(), OutputFrameSeries::writeFloat(), OutputFrameSeries::writeGray(), and OutputFrameSeries::writeRGB().

### 10.97.4 Member Data Documentation

#### 10.97.4.1 **float FrameRange::delay**

inter-frame delay in ms

Definition at line 55 of file FrameRange.H.

Referenced by convertFromString(), convertToString(), FrameSeries::delay(), and FrameSeries::update().

#### 10.97.4.2 **int FrameRange::first**

number of first frame

Definition at line 53 of file FrameRange.H.

Referenced by FrameSeries::computeFileName(), convertFromString(), convertToString(), InputFrameSeries::getImageDims(), isMultiframe(), FrameSeries::minframe(), FrameSeries::reset(), FrameSeries::start2(), and FrameSeries::update().

#### 10.97.4.3 **int FrameRange::last**

number of last frame

Definition at line 54 of file FrameRange.H.

Referenced by `FrameSeries::computeFileName()`, `convertFromString()`, `convertToString()`, `FrameSeries::finished()`, `isMultiframe()`, and `FrameSeries::maxframe()`.

The documentation for this class was generated from the following file:

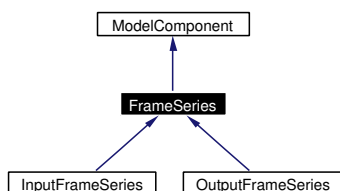
- [FrameRange.H](#)

## 10.98 FrameSeries Class Reference

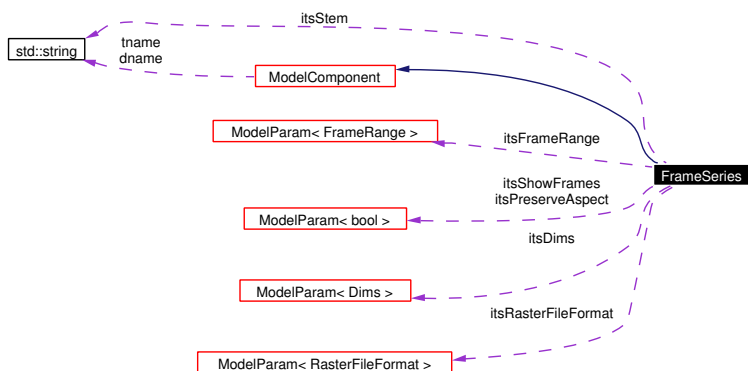
FrameSeries manages a series of input or output frames.

```
#include <FrameSeries.H>
```

Inheritance diagram for FrameSeries:



Collaboration diagram for FrameSeries:



### Access functions

- enum `State` { `SAME`, `NEXT`, `COMPLETE` }  
The possible states returned by `update()`.
- `std::string getFileStem () const`  
Get the file stem.
- `State update (double stime, double sdelay)`  
Update internal state and return corresponding `FrameSeries::State`.

- [State updateNext \(\)](#)  
*Update by moving to the next frame and not worrying about time counts.*
- `bool` [isMultiframe \(\)](#) `const`  
*Returns true if the series has multiple frames.*
- `double` [delay \(\)](#) `const`  
*Returns the inter-frame delay in seconds.*
- `int` [frame \(\)](#) `const`  
*Returns the current frame.*
- `int` [minframe \(\)](#) `const`  
*Returns the first frame in the series.*
- `int` [maxframe \(\)](#) `const`  
*Returns the last valid frame in the series.*
- `int` [numFrames \(\)](#) `const`  
*Returns the number of frames in the series.*
- `bool` [finished \(\)](#) `const`  
*True if reached beyond last frame in series.*
- `bool` [didDisplayFrames \(\)](#) `const`  
*Did we display any frames since last update?*
- `template<class T> Image< T > resizeImage (const Image< T > &input)`  
`const`  
*resize an image to our dims*

## Public Types

## Public Member Functions

### Constructors, destructors and basic setup

- `FrameSeries (ModelManager &mgr, const std::string &prefixName, const std::string &descrName="Frame Series", const std::string &tagName="FrameSeries")`  
*Constructor.*

- virtual void [exportOptions](#) (const int whatToExport, const bool recurse=true)  
*Export some of our [ModelParam](#) parameters as command-line-options.*
- virtual [~FrameSeries](#) ()  
*Destructor.*
- virtual void [reset](#) (const bool recurse=true)  
*resets this FrameSeries to its state after construction*
- void [setFileStem](#) (const [std::string](#) &stem)  
*Set the file stem - for more info see [ModelComponent](#).*

### Protected Member Functions

- [std::string computeFileName](#) (const int framenum=-1, const char \*otherstem=0, va\_list ap=0) const  
*compute filename corresponding to args and return as string*
- bool [okToDisplay](#) ()  
*have we displayed too many windows, or can we still display more?*
- virtual void [start2](#) ()  
*post-commandline-parsing initialization (see [ModelComponent.H](#))*

### Protected Attributes

- [ModelParam< FrameRange > itsFrameRange](#)  
*range of frames and inter-frame delay in SECONDS*
- [ModelParam< Dims > itsDims](#)  
*dims to which frames should be resized on read/write, or 0x0 for none*
- [ModelParam< bool > itsPreserveAspect](#)  
*if true, aspect ratio of original image will be preserved when resizing*
- [ModelParam< bool > itsShowFrames](#)  
*show the frames using xv*
- [ModelParam< RasterFileFormat > itsRasterFileFormat](#)  
*default file format; will be used if Auto is specified at read/write*



- [std::string itsStem](#)  
*filename stem*
- [int itsCurrentFrame](#)  
*current frame number*
- [double itsOldTime](#)  
*time of last [update\(\)](#)*
- [int itsDidDisplay](#)  
*did we show some frames since last update?*

### 10.98.1 Detailed Description

FrameSeries manages a series of input or output frames.

It keeps track of a range of frames ([first..last]) and an interframe delay in seconds (see [FrameRange.H](#)). Its key behavior is the [update\(\)](#) function, which takes the current simulation time in seconds and the simulation step in s, and returns a status code indicating whether the frame series (a) was unchanged, (b) will require that the next frame be loaded just before the simulation time is advanced by the current simulation step, or (c) is finished. [InputFrameSeries](#) and [OutputFrameSeries](#) derive from FrameSeries and implement a number of helper functions to facilitate reading/writing frames in a standardized manner, with possible on-the-fly rescaling and visualization. NOTE: if our [ModelManager](#) is in test mode (manager.testMode() returns true, typically because `-test-mode` was given on the command line), we will turn off all displays even though they may be on here.

Definition at line 66 of file FrameSeries.H.

### 10.98.2 Member Enumeration Documentation

#### 10.98.2.1 [enum FrameSeries::State](#)

The possible states returned by [update\(\)](#).

Definition at line 104 of file FrameSeries.H.

Referenced by [update\(\)](#), and [updateNext\(\)](#).

### 10.98.3 Constructor & Destructor Documentation

#### 10.98.3.1 `FrameSeries::FrameSeries (ModelManager & mgr, const std::string & prefixName, const std::string & descrName = "Frame Series", const std::string & tagName = "FrameSeries")`

Constructor.

**Parameters:**

*prefixName* should be "Input" or "Output" and will be automatically provided by the constructors of [InputFrameSeries](#) and [OutputFrameSeries](#)

Definition at line 52 of file `FrameSeries.C`.

References `RasterFileFormat`.

#### 10.98.3.2 `FrameSeries::~~FrameSeries () [virtual]`

Destructor.

Definition at line 89 of file `FrameSeries.C`.

### 10.98.4 Member Function Documentation

#### 10.98.4.1 `std::string FrameSeries::computeFileName (const int framenum = -1, const char * otherstem = 0, va_list ap = 0) const [protected]`

compute filename corresponding to args and return as string

Definition at line 185 of file `FrameSeries.C`.

References `FrameRange::first`, `ModelParam< FrameRange >::getVal()`, `FrameRange::isMultiframe()`, `itsCurrentFrame`, `itsFrameRange`, `itsStem`, `FrameRange::last`, `ModelComponent::tagName()`, and `uint`.

Referenced by `InputFrameSeries::getImageDims()`, `InputFrameSeries::readGray()`, `InputFrameSeries::readRGB()`, `OutputFrameSeries::writeFloat()`, `OutputFrameSeries::writeGray()`, and `OutputFrameSeries::writeRGB()`.

#### 10.98.4.2 `double FrameSeries::delay () const`

Returns the inter-frame delay in seconds.

Definition at line 121 of file `FrameSeries.C`.

References `FrameRange::delay`, `ModelParam< FrameRange >::getVal()`, and `itsFrameRange`.

#### 10.98.4.3 `bool FrameSeries::didDisplayFrames () const`

Did we display any frames since last update?

This is reset to false on each call to [update\(\)](#). It is set to true each time a frame is displayed (if `itsShowFrames` is turned on). You can check it in order to decide whether you should pause and wait for the user after each series of displays at a given time step).

Definition at line 181 of file `FrameSeries.C`.

References `itsDidDisplay`.

#### 10.98.4.4 `void FrameSeries::exportOptions (const int whatToExport, const bool recurse = true) [virtual]`

Export some of our [ModelParam](#) parameters as command-line-options.

See the base function in [ModelComponent.H](#) for info.

Reimplemented from [ModelComponent](#).

Definition at line 66 of file `FrameSeries.C`.

References `ModelComponent::exportOptions()`, `itsFrameRange`, `itsPreserveAspect`, `itsRasterFileFormat`, `itsShowFrames`, `OPTEXP_CORE`, and `ModelManager::requestOption()`.

#### 10.98.4.5 `bool FrameSeries::finished () const`

True if reached beyond last frame in series.

Definition at line 141 of file `FrameSeries.C`.

References `ModelParam< FrameRange >::getVal()`, `itsCurrentFrame`, `itsFrameRange`, and `FrameRange::last`.

Referenced by `update()`, and `updateNext()`.

#### 10.98.4.6 `int FrameSeries::frame () const`

Returns the current frame.

Definition at line 125 of file `FrameSeries.C`.

References `itsCurrentFrame`.

**10.98.4.7 `std::string` FrameSeries::getFileStem () const**

Get the file stem.

Definition at line 113 of file FrameSeries.C.

References itsStem.

**10.98.4.8 `bool` FrameSeries::isMultiframe () const**

Returns true if the series has multiple frames.

Definition at line 117 of file FrameSeries.C.

References `ModelParam< FrameRange >::getVal()`, `FrameRange::isMultiframe()`, and `itsFrameRange`.

Referenced by `update()`.

**10.98.4.9 `int` FrameSeries::maxframe () const**

Returns the last valid frame in the series.

Definition at line 133 of file FrameSeries.C.

References `ModelParam< FrameRange >::getVal()`, `itsFrameRange`, and `FrameRange::last`.

Referenced by `numFrames()`.

**10.98.4.10 `int` FrameSeries::minframe () const**

Returns the first frame in the series.

Definition at line 129 of file FrameSeries.C.

References `FrameRange::first`, `ModelParam< FrameRange >::getVal()`, and `itsFrameRange`.

Referenced by `numFrames()`.

**10.98.4.11 `int` FrameSeries::numFrames () const**

Returns the number of frames in the series.

Definition at line 137 of file FrameSeries.C.

References `maxframe()`, and `minframe()`.

**10.98.4.12 bool FrameSeries::okToDisplay () [protected]**

have we displayed too many windows, or can we still display more?

Definition at line 268 of file FrameSeries.C.

References `ModelParam< bool >::getVal()`, `itsDidDisplay`, `itsShowFrames`, and `ModelManager::testMode()`.

Referenced by `InputFrameSeries::readGray()`, `InputFrameSeries::readRGB()`, `OutputFrameSeries::writeFloat()`, `OutputFrameSeries::writeGray()`, and `OutputFrameSeries::writeRGB()`.

**10.98.4.13 void FrameSeries::reset (const bool *recurse* = true) [virtual]**

resets this FrameSeries to its state after construction

Reimplemented from [ModelComponent](#).

Definition at line 93 of file FrameSeries.C.

References `FrameRange::first`, `ModelParam< FrameRange >::getVal()`, `itsCurrentFrame`, `itsDidDisplay`, `itsFrameRange`, `itsOldTime`, and `ModelComponent::reset()`.

**10.98.4.14 template<class T> [Image](#)< T > FrameSeries::resizeImage (const [Image](#)< T > & *input*) const**

resize an image to our dims

Definition at line 253 of file FrameSeries.C.

References `Image< T >::embed()`, `Image< T >::getHeight()`, `ModelParam< bool >::getVal()`, `ModelParam< Dims >::getVal()`, `Image< T >::getWidth()`, `Dims::isEmpty()`, `itsPreserveAspect`, `rescale()`, and `ZEROS`.

Referenced by `InputFrameSeries::readGray()`, `InputFrameSeries::readRGB()`, `OutputFrameSeries::writeFloat()`, `OutputFrameSeries::writeGray()`, and `OutputFrameSeries::writeRGB()`.

**10.98.4.15 void FrameSeries::setFileStem (const [std::string](#) & *stem*)**

Set the file stem - for more info see [ModelComponent](#).

Definition at line 105 of file FrameSeries.C.

References `itsStem`, and `ModelComponent::started()`.

**10.98.4.16 void FrameSeries::start2 ()** [protected, virtual]

post-commandline-parsing initialization (see [ModelComponent.H](#))

Reimplemented from [ModelComponent](#).

Reimplemented in [InputFrameSeries](#).

Definition at line 82 of file FrameSeries.C.

References [FrameRange::first](#), [ModelParam< FrameRange >::getVal\(\)](#), [itsCurrentFrame](#), [itsFrameRange](#), and [itsOldTime](#).

Referenced by [InputFrameSeries::start2\(\)](#).

**10.98.4.17 [FrameSeries::State](#) FrameSeries::update (double *stime*, double *sdelay*)**

Update internal state and return corresponding [FrameSeries::State](#).

**Parameters:**

*stime* current time in seconds

*sdelay* time step in seconds

Definition at line 145 of file FrameSeries.C.

References [FrameRange::delay](#), [ModelComponent::descriptiveName\(\)](#), [finished\(\)](#), [FrameRange::first](#), [ModelParam< FrameRange >::getVal\(\)](#), [isMultiframe\(\)](#), [FrameRange::isMultiframe\(\)](#), [itsCurrentFrame](#), [itsDidDisplay](#), [itsFrameRange](#), [itsOldTime](#), [State](#), and [ModelManager::textLog\(\)](#).

**10.98.4.18 [FrameSeries::State](#) FrameSeries::updateNext ()**

Update by moving to the next frame and not worrying about time counts.

Definition at line 173 of file FrameSeries.C.

References [finished\(\)](#), [itsCurrentFrame](#), and [State](#).

**10.98.5 Member Data Documentation****10.98.5.1 int [FrameSeries::itsCurrentFrame](#)** [protected]

current frame number

Definition at line 164 of file FrameSeries.H.

Referenced by [computeFileName\(\)](#), [finished\(\)](#), [frame\(\)](#), [reset\(\)](#), [start2\(\)](#), [update\(\)](#), and [updateNext\(\)](#).

**10.98.5.2** `int` [FrameSeries::itsDidDisplay](#) [protected]

did we show some frames since last update?

Definition at line 166 of file FrameSeries.H.

Referenced by [didDisplayFrames\(\)](#), [okToDisplay\(\)](#), [reset\(\)](#), and [update\(\)](#).

**10.98.5.3** `ModelParam<Dims>` [FrameSeries::itsDims](#) [protected]

dims to which frames should be resized on read/write, or 0x0 for none

Definition at line 152 of file FrameSeries.H.

**10.98.5.4** `ModelParam<FrameRange>` [FrameSeries::itsFrameRange](#) [protected]

range of frames and inter-frame delay in SECONDS

Definition at line 149 of file FrameSeries.H.

Referenced by [computeFileName\(\)](#), [delay\(\)](#), [exportOptions\(\)](#), [finished\(\)](#), [is-Multiframe\(\)](#), [maxframe\(\)](#), [minframe\(\)](#), [reset\(\)](#), [start2\(\)](#), and [update\(\)](#).

**10.98.5.5** `double` [FrameSeries::itsOldTime](#) [protected]

time of last [update\(\)](#)

Definition at line 165 of file FrameSeries.H.

Referenced by [reset\(\)](#), [start2\(\)](#), and [update\(\)](#).

**10.98.5.6** `ModelParam<bool>` [FrameSeries::itsPreserveAspect](#) [protected]

if true, aspect ratio of original image will be preserved when resizing

Definition at line 155 of file FrameSeries.H.

Referenced by [exportOptions\(\)](#), and [resizeImage\(\)](#).

**10.98.5.7** `ModelParam<RasterFileFormat>` [FrameSeries::itsRasterFileFormat](#) [protected]

default file format; will be used if Auto is specified at read/write

Definition at line 161 of file FrameSeries.H.

Referenced by `exportOptions()`.

#### **10.98.5.8**   [ModelParam<bool>](#) [FrameSeries::itsShowFrames](#)   [protected]

show the frames using xv

Definition at line 158 of file `FrameSeries.H`.

Referenced by `exportOptions()`, and `okToDisplay()`.

#### **10.98.5.9**   [std::string](#) [FrameSeries::itsStem](#)   [protected]

filename stem

Definition at line 163 of file `FrameSeries.H`.

Referenced by `computeFileName()`, `getFileStem()`, and `setFileStem()`.

The documentation for this class was generated from the following files:

- [FrameSeries.H](#)
- [FrameSeries.C](#)

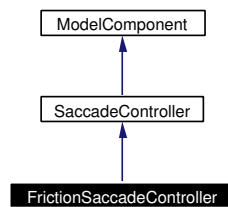


## 10.99 FrictionSaccadeController Class Reference

Friction-damped mass-spring saccade controller.

```
#include <SaccadeControllers.H>
```

Inheritance diagram for FrictionSaccadeController:



Collaboration diagram for FrictionSaccadeController:

### Public Member Functions

- [FrictionSaccadeController](#) ([ModelManager](#) &mgr, const [std::string](#) paramprefix="SCeye")  
*Constructor.*
- virtual [~FrictionSaccadeController](#) ()  
*Destructor.*
- virtual void [exportOptions](#) (const int whatToExport, const bool recurse=true)  
*Export some of our [ModelParam](#) parameters as command-line-options.*
- virtual void [evolve](#) (const double t)  
*evolve one time step (called, e.g., by [Brain::evolve\(\)](#))*

### Protected Member Functions

- virtual void [computeWhenNewPercept](#) ()  
*This method is called each time a new percept has arrived.*
- virtual void [computeWhenResetPos](#) ()  
*This method is called each time a reset is made.*

- virtual [Point2D](#) [computeWhenNewDecision](#) (const double t)  
*This method is called each time a new decision is requested.*
- virtual void [start1](#) ()  
*get us started:*

## Protected Attributes

- [ModelParam](#)< double > [itsSpringK](#)  
*spring stiffness*
- [ModelParam](#)< double > [itsFrictionMu](#)  
*friction coeff*
- [ModelParam](#)< bool > [itsStartAtIEP](#)  
*start at initial eye position?*
- [ModelParam](#)< [Point2D](#) > [itsIEP](#)  
*Initial (and permanent) eye position.*
- [ModelParam](#)< [Dims](#) > [itsDims](#)  
*Image size so that we stay inside it.*
- [ModelParam](#)< double > [itsTimeStep](#)  
*Simulation time step.*

### 10.99.1 Detailed Description

Friction-damped mass-spring saccade controller.

FrictionSaccadeController acts like a mass (at saccade center) connected to the covert focus of attention through a spring of zero rest length, and sliding on a plane in the presence of friction.

Definition at line 183 of file SaccadeControllers.H.

### 10.99.2 Constructor & Destructor Documentation

#### 10.99.2.1 FrictionSaccadeController::FrictionSaccadeController ([ModelManager](#) & mgr, const [std::string](#) paramprefix = "SCeye")

Constructor.

**Parameters:**

*paramprefix* As this controller may be used either for the eye or for the head (as part of bigger controllers that include both eye and head), this should be either "SCeye" or "SChead" and will determine which model options we use for our internal parameters (eye or head)

Definition at line 247 of file SaccadeControllers.C.

**10.99.2.2 FrictionSaccadeController::~FrictionSaccadeController ()**  
[virtual]

Destructor.

Definition at line 274 of file SaccadeControllers.C.

**10.99.3 Member Function Documentation****10.99.3.1 Point2D FrictionSaccadeController::computeWhenNewDecision**  
(const double t) [protected, virtual]

This method is called each time a new decision is requested.

Implements [SaccadeController](#).

Definition at line 367 of file SaccadeControllers.C.

References [ModelParam< Dims >::getVal\(\)](#), [Dims::h\(\)](#), [SaccadeController::have-Percepts\(\)](#), [Point2D::i](#), [Point2D::j](#), and [Dims::w\(\)](#).

**10.99.3.2 void FrictionSaccadeController::computeWhenNewPercept ()**  
[protected, virtual]

This method is called each time a new percept has arrived.

Implements [SaccadeController](#).

Definition at line 338 of file SaccadeControllers.C.

References [SaccadeController::getPreviousPercept\(\)](#), [Point2D::i](#), [Point2D::j](#), [WTAwinner::p](#), and [WTAwinner::t](#).

**10.99.3.3 void FrictionSaccadeController::computeWhenResetPos ()**  
[protected, virtual]

This method is called each time a reset is made.

Implements [SaccadeController](#).

Definition at line 351 of file SaccadeControllers.C.

References `SaccadeController::beginSaccade()`, `SaccadeController::getPreviousPercept()`, `Point2D::i`, `Point2D::j`, `WTAwinner::p`, and `WTAwinner::t`.

#### 10.99.3.4 `void FrictionSaccadeController::evolve (const double t) [virtual]`

evolve one time step (called, e.g., by `Brain::evolve()`)

Implements `SaccadeController`.

Definition at line 296 of file SaccadeControllers.C.

References `SaccadeController::endSaccade()`, `SaccadeController::getPreviousPercept()`, `ModelParam< double >::getVal()`, `SaccadeController::havePercepts()`, `Point2D::i`, `SaccadeController::isInSaccade()`, `itsFrictionMu`, `itsSpringK`, `Point2D::j`, and `WTAwinner::p`.

#### 10.99.3.5 `void FrictionSaccadeController::exportOptions (const int whatToExport, const bool recurse = true) [virtual]`

Export some of our `ModelParam` parameters as command-line-options.

See the base function in `ModelComponent.H` for info.

Reimplemented from `ModelComponent`.

Definition at line 278 of file SaccadeControllers.C.

References `ModelComponent::exportOptions()`, `itsFrictionMu`, `itsSpringK`, `itsStartAtIEP`, `OPTEXP_CORE`, and `ModelManager::requestOption()`.

#### 10.99.3.6 `void FrictionSaccadeController::start1 () [protected, virtual]`

get us started:

Reimplemented from `ModelComponent`.

Definition at line 262 of file SaccadeControllers.C.

References `ModelParam< Point2D >::getVal()`, `ModelParam< bool >::getVal()`, `itsStartAtIEP`, `WTAwinner::p`, and `SaccadeController::setPercept()`.

## 10.99.4 Member Data Documentation

### 10.99.4.1 [ModelParam<Dims>](#) [FrictionSaccadeController::itsDims](#) [protected]

[Image](#) size so that we stay inside it.

Definition at line 221 of file SaccadeControllers.H.

### 10.99.4.2 [ModelParam<double>](#) [FrictionSaccadeController::itsFrictionMu](#) [protected]

friction coeff

Definition at line 218 of file SaccadeControllers.H.

Referenced by [evolve\(\)](#), and [exportOptions\(\)](#).

### 10.99.4.3 [ModelParam<Point2D>](#) [FrictionSaccadeController::itsIEP](#) [protected]

Initial (and permanent) eye position.

Definition at line 220 of file SaccadeControllers.H.

### 10.99.4.4 [ModelParam<double>](#) [FrictionSaccadeController::itsSpringK](#) [protected]

spring stiffness

Definition at line 217 of file SaccadeControllers.H.

Referenced by [evolve\(\)](#), and [exportOptions\(\)](#).

### 10.99.4.5 [ModelParam<bool>](#) [FrictionSaccadeController::itsStartAtIEP](#) [protected]

start at initial eye position?

Definition at line 219 of file SaccadeControllers.H.

Referenced by [exportOptions\(\)](#), and [start1\(\)](#).

#### 10.99.4.6 [ModelParam<double>](#) [FrictionSaccadeController::itsTimeStep](#) [protected]

Simulation time step.

Definition at line 222 of file SaccadeControllers.H.

The documentation for this class was generated from the following files:

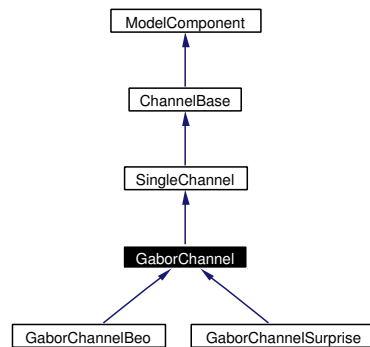
- [SaccadeControllers.H](#)
- [SaccadeControllers.C](#)

## 10.100 GaborChannel Class Reference

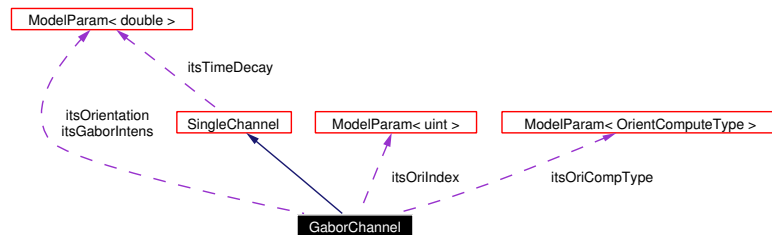
An orientation-tuned gabor channel.

```
#include <Channels.H>
```

Inheritance diagram for GaborChannel:



Collaboration diagram for GaborChannel:



### Public Member Functions

- **GaborChannel** (**ModelManager** &mgr, const **uint** oriIndex, const double ang)  
*Construct with standard params plus an orientation index for this object.*
- virtual **~GaborChannel** ()  
*Destructor.*
- virtual void **exportOptions** (const int whatToExport, const bool recurse=true)  
*Export some of our [ModelParam](#) parameters as command-line-options.*

- virtual void [readFrom](#) (const [ParamMap](#) &pmap)  
*Read params from the [ParamMap](#).*
- virtual void [writeTo](#) ([ParamMap](#) &pmap) const  
*Write params to the [ParamMap](#).*
- double [angle](#) () const  
*Get the preferred orientation (in degrees) of this channel.*

### Protected Member Functions

- virtual void [start1](#) ()
- void [resetPyramid](#) ()

### Protected Attributes

- [ModelParam](#)< uint > [itsOriIndex](#)
- [ModelParam](#)< double > [itsOrientation](#)
- [ModelParam](#)< double > [itsGaborIntens](#)
- [ModelParam](#)< [OrientComputeType](#) > [itsOriCompType](#)  
*determines the type of computation used to compute orientation*

#### 10.100.1 Detailed Description

An orientation-tuned gabor channel.

Definition at line 231 of file Channels.H.

#### 10.100.2 Constructor & Destructor Documentation

##### 10.100.2.1 [GaborChannel::GaborChannel](#) ([ModelManager](#) & mgr, const uint [oriIndex](#), const double [ang](#))

Construct with standard params plus an orientation index for this object.

##### Parameters:

**[oriIndex](#)** is used for various indexing such as with [ParamMap](#) and [Jet](#) interfacing.  
**[ang](#)** is the preferred orientation for this channel, in degrees (0..180).



Definition at line 279 of file Channels.C.

References [ORI](#), [OrientComputeType](#), [ORISteerable](#), [ModelComponent::setDescriptiveName\(\)](#), [ModelComponent::setTagName\(\)](#), [ModelParam< bool >::setVal\(\)](#), and [uint](#).

### 10.100.2.2 GaborChannel::~GaborChannel () [virtual]

Destructor.

Definition at line 305 of file Channels.C.

## 10.100.3 Member Function Documentation

### 10.100.3.1 double GaborChannel::angle () const

Get the preferred orientation (in degrees) of this channel.

Definition at line 354 of file Channels.C.

References [ModelParam< double >::getVal\(\)](#).

Referenced by [SoxChannel::getNonlinearResponse\(\)](#), and [main\(\)](#).

### 10.100.3.2 void GaborChannel::exportOptions (const int *whatToExport*, const bool *recurse* = true) [virtual]

Export some of our [ModelParam](#) parameters as command-line-options.

See the base function in [ModelComponent.H](#) for info.

Reimplemented from [SingleChannel](#).

Definition at line 309 of file Channels.C.

References [SingleChannel::exportOptions\(\)](#), [itsOriCompType](#), [OPTEXP\\_CORE](#), and [ModelManager::requestOption\(\)](#).

### 10.100.3.3 void GaborChannel::readFrom (const [ParamMap](#) & *pmap*) [virtual]

Read params from the [ParamMap](#).

Reimplemented from [SingleChannel](#).

Definition at line 322 of file Channels.C.

References [ModelParam< OrientComputeType >::getVal\(\)](#), [ModelParam< double >::getVal\(\)](#), [itsOriCompType](#), [OrientComputeType](#), [ParamMap::queryDouble-](#)

`Param()`, `ParamMap::queryIntParam()`, `SingleChannel::readFrom()`, `ModelParam< OrientComputeType >::setVal()`, and `ModelParam< double >::setVal()`.

#### 10.100.3.4 `void GaborChannel::start1 ()` [protected, virtual]

If you overload this, make sure you call [SingleChannel::start1\(\)](#) at the beginning of your overload

Reimplemented from [SingleChannel](#).

Reimplemented in [GaborChannelBeo](#).

Definition at line 298 of file `Channels.C`.

References `SingleChannel::start1()`.

Referenced by `GaborChannelBeo::start1()`.

#### 10.100.3.5 `void GaborChannel::writeTo (ParamMap & pmap) const` [virtual]

Write params to the [ParamMap](#).

Reimplemented from [SingleChannel](#).

Definition at line 346 of file `Channels.C`.

References `ModelParam< OrientComputeType >::getVal()`, `ModelParam< double >::getVal()`, `itsOriCompType`, `ParamMap::putDoubleParam()`, `ParamMap::putIntParam()`, and `SingleChannel::writeTo()`.

### 10.100.4 Member Data Documentation

#### 10.100.4.1 [ModelParam<OrientComputeType>](#) [GaborChannel::itsOriCompType](#) [protected]

determines the type of computation used to compute orientation

see `OrientComputeTypes.H` for possible values.

Definition at line 263 of file `Channels.H`.

Referenced by `exportOptions()`, `readFrom()`, and `writeTo()`.

The documentation for this class was generated from the following files:

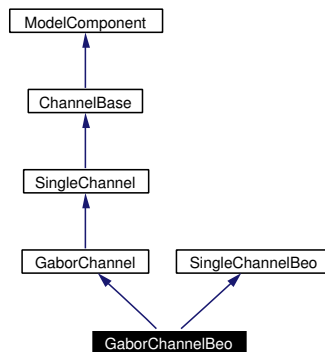
- [Channels.H](#)
- [Channels.C](#)

## 10.101 GaborChannelBeo Class Reference

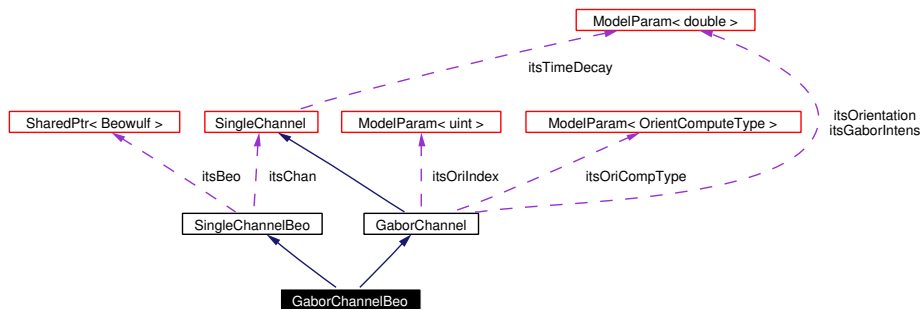
Beowulf version of [GaborChannel](#).

```
#include <ChannelsBeo.H>
```

Inheritance diagram for GaborChannelBeo:



Collaboration diagram for GaborChannelBeo:



### Public Member Functions

- **GaborChannelBeo** ([ModelManager](#) &mgr, const [uint](#) oriIndex, const double ang, const [SharedPtr< Beowulf >](#) &beow)
- virtual bool [outputAvailable](#) () const

*Query whether the channel is ready to give yield valid output.*

## Protected Member Functions

- virtual void `doInput` (const `Image`< `PixRGB`< float > > \*, const `Image`< float > \* `bwimg`, const double `t`, const `Image`< byte > &`clipMask`)  
*`SingleChannel` requires only luminance input.*
- virtual void `start1` ()

### 10.101.1 Detailed Description

`Beowulf` version of `GaborChannel`.

Definition at line 97 of file `ChannelsBeo.H`.

### 10.101.2 Member Function Documentation

**10.101.2.1 void `GaborChannelBeo::doInput` (const `Image`< `PixRGB`< float > > \*, const `Image`< float > \* `bwimg`, const double `t`, const `Image`< byte > &`clipMask`)** [protected, virtual]

`SingleChannel` requires only luminance input.

Reimplemented from `SingleChannel`.

Definition at line 133 of file `ChannelsBeo.C`.

References `SingleChannelBeo::sendInput()`.

**10.101.2.2 bool `GaborChannelBeo::outputAvailable` () const** [virtual]

Query whether the channel is ready to give yield valid output.

In some cases (e.g. with motion channels), it may take several input frames before the channel is fully initialized.

Reimplemented from `SingleChannel`.

Definition at line 126 of file `ChannelsBeo.C`.

References `SingleChannel::outputAvailable()`.

**10.101.2.3 void `GaborChannelBeo::start1` ()** [protected, virtual]

If you overload this, make sure you call `SingleChannel::start1()` at the beginning of your overload

Reimplemented from `GaborChannel`.

Definition at line 137 of file ChannelsBeo.C.

References `SingleChannelBeo::requestBeowulfNode()`, and `GaborChannel::start1()`.

The documentation for this class was generated from the following files:

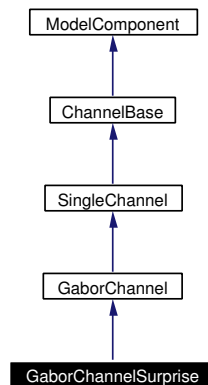
- [ChannelsBeo.H](#)
- [ChannelsBeo.C](#)

## 10.102 GaborChannelSurprise Class Reference

An orientation-tuned gabor channel.

```
#include <ChannelsSurprise.H>
```

Inheritance diagram for GaborChannelSurprise:



Collaboration diagram for GaborChannelSurprise:

### Public Member Functions

- [GaborChannelSurprise](#) ([ModelManager](#) &mgr, const [uint](#) oriIndex, const double ang)  
*Construct with standard params plus an orientation index for this object.*
- virtual [~GaborChannelSurprise](#) ()  
*Destructor.*
- virtual [Image](#)< float > [computeSubmap](#) (const [uint](#) index) const  
*[computeSubmap\(\)](#) is special and will return surprise*

### Protected Attributes

- [SharedPtr](#)< [SingleChannelSurprise](#) > itsSCS  
*our surprise computer*

### 10.102.1 Detailed Description

An orientation-tuned gabor channel.

Definition at line 267 of file ChannelsSurprise.H.

### 10.102.2 Constructor & Destructor Documentation

#### 10.102.2.1 GaborChannelSurprise::GaborChannelSurprise ([ModelManager](#) & *mgr*, const [uint](#) *oriIndex*, const double *ang*)

Construct with standard params plus an orientation index for this object.

Definition at line 244 of file ChannelsSurprise.C.

References [ModelComponent::addSubComponent\(\)](#), and [uint](#).

#### 10.102.2.2 GaborChannelSurprise::~~GaborChannelSurprise () [virtual]

Destructor.

Definition at line 252 of file ChannelsSurprise.C.

### 10.102.3 Member Function Documentation

#### 10.102.3.1 [Image](#)< float > GaborChannelSurprise::computeSubmap (const [uint](#) *index*) const [virtual]

[computeSubmap\(\)](#) is special and will return surprise

Reimplemented from [SingleChannel](#).

Definition at line 255 of file ChannelsSurprise.C.

References [uint](#).

### 10.102.4 Member Data Documentation

#### 10.102.4.1 [SharedPtr](#)<[SingleChannelSurprise](#)> GaborChannelSurprise::its-SCS [protected]

our surprise computer

Definition at line 281 of file ChannelsSurprise.H.

The documentation for this class was generated from the following files:

- [ChannelsSurprise.H](#)
- ChannelsSurprise.C

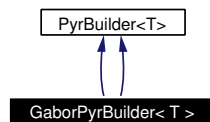


## 10.103 GaborPyrBuilder< T > Class Template Reference

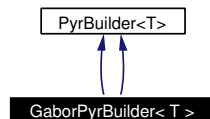
A pyramid based on convolution with oriented Gabor filters.

```
#include <GaborPyramid.H>
```

Inheritance diagram for GaborPyrBuilder< T >:



Collaboration diagram for GaborPyrBuilder< T >:



### Public Member Functions

- `GaborPyrBuilder` (double angle, double filter\_period, double elongation=1.0)  
*Constructor.*
- virtual `GaborPyrBuilder< T > * clone ()` const  
*Cloning constructor.*
- virtual `ImageSet< T > build` (const `Image< T > &img`, const int depth)  
*Builds a Gabor pyramid based on img.*
- `ImageSet< T > input` (const `ImageSet< T > &pyr`)  
*Builds a Gabor pyramid from the levels in the given pyramid.*
- `GaborPyrBuilder` (double angle, double filter\_period, double elongation=1.0, int size=-1, int buildFlags=0)  
*Constructor.*
- virtual `GaborPyrBuilder< T > * clone ()` const

*Cloning constructor.*

- virtual `ImageSet< T > build` (const `Image< T > &img`, const int depth)  
*Builds a Gabor pyramid based on an image.*
- `ImageSet< T > input` (const `ImageSet< T > &pyr`)  
*Builds a Gabor pyramid from the levels in the given pyramid.*

### 10.103.1 Detailed Description

`template<class T> class GaborPyrBuilder< T >`

A pyramid based on convolution with oriented Gabor filters.

Definition at line 61 of file GaborPyramid.H.

### 10.103.2 Constructor & Destructor Documentation

10.103.2.1 `template<class T> GaborPyrBuilder< T >::GaborPyrBuilder`  
(double *angle*, double *filter\_period*, double *elongation* = 1.0)

Constructor.

**Parameters:**

- angle* Angle (in degrees); parallel to the Gabor's "ridges".
- filter\_period* Period (in pixels) of underlying grating.
- elongation* Ratio of major/minor axis lengths of Gaussian envelope.

Definition at line 262 of file GaborPyramid.C.

10.103.2.2 `template<class T> GaborPyrBuilder< T >::GaborPyrBuilder`  
(double *angle*, double *filter\_period*, double *elongation* = 1.0, int *size* = -1, int *buildFlags* = 0)

Constructor.

**Parameters:**

- angle* Angle (in degrees); parallel to the Gabor's "ridges".
- filter\_period* Period (in pixels) of underlying grating.
- elongation* Ratio of major/minor axis lengths of Gaussian envelope.

*size* determines the filter size (in pixels) - in the default case (-1), the size is computed from the filter\_preiod and elongation

*buildFlags* see [Pyramid\\_Ops.H](#) for possible values

Definition at line 336 of file PyrBuilder.C.

### 10.103.3 Member Function Documentation

**10.103.3.1** `template<class T> virtual ImageSet<T> GaborPyrBuilder< T >::build (const Image< T > & img, const int depth) [virtual]`

Builds a Gabor pyramid based on an image.

Implements [PyrBuilder< T >](#).

**10.103.3.2** `template<class T> ImageSet< T > GaborPyrBuilder< T >::build (const Image< T > & img, const int depth) [virtual]`

Builds a Gabor pyramid based on *img*.

Implements [PyrBuilder< T >](#).

Definition at line 278 of file GaborPyramid.C.

References [buildPyrGabor\(\)](#).

**10.103.3.3** `template<class T> virtual GaborPyrBuilder<T>* GaborPyrBuilder< T >::clone () const [virtual]`

Cloning constructor.

Implements [PyrBuilder< T >](#).

**10.103.3.4** `template<class T> GaborPyrBuilder< T > * GaborPyrBuilder< T >::clone () const [virtual]`

Cloning constructor.

Implements [PyrBuilder< T >](#).

Definition at line 272 of file GaborPyramid.C.

**10.103.3.5** `template<class T> ImageSet<T> GaborPyrBuilder< T >::input (const ImageSet< T > & pyr)`

Builds a Gabor pyramid from the levels in the given pyramid.

**10.103.3.6** `template<class T> ImageSet< T > GaborPyrBuilder< T >::input  
(const ImageSet< T > & pyr)`

Builds a Gabor pyramid from the levels in the given pyramid.

Definition at line 284 of file `GaborPyramid.C`.

References `buildPyrGabor()`.

The documentation for this class was generated from the following files:

- [GaborPyramid.H](#)
- [PyrBuilder.H](#)
- [GaborPyramid.C](#)
- [PyrBuilder.C](#)

## 10.104 GACHromosome Class Reference

Chromosome class for genetic algorithm.

```
#include <GACHromosome.H>
```

### Public Member Functions

#### Constructor, destructor and initialization.

- [GACHromosome](#) ()  
*Uninitialized constructor. Need to call [resize\(\)](#) later.*
- [GACHromosome](#) (const int N, const int \*a)  
*Construct from C array.*
- [GACHromosome](#) (const int N)  
*Construct random chromosome.*
- [GACHromosome](#) (const [GACHromosome](#) &c)  
*Copy constructor.*
- void [resize](#) (const int N)  
*Change size, do not initialize.*
- void [init](#) (const int N, const int \*a)  
*Initialize from C array.*
- void [init](#) (const int N)  
*Initialize randomly.*
- [~GACHromosome](#) ()  
*Destructor.*

#### Access functions

- int [get\\_size](#) () const  
*Return the size of the chromosome.*
- void [set\\_gene](#) (const int i, const int a)  
*Set the gene #i to a.*
- int [get\\_gene](#) (const int i) const  
*Return the gene #i.*

- void [set\\_fitness](#) (const float a)  
*Set the fitness to a.*
- float [get\\_fitness](#) () const  
*Return the fitness.*
- void [set\\_linear\\_fitness](#) (const float a)  
*Set the linear fitness to a.*
- float [get\\_linear\\_fitness](#) () const  
*Return the linear fitness.*
- void [set\\_breedings](#) (const int a)  
*Set breedings to a.*
- int [get\\_breedings](#) () const  
*Return the breedings.*

### Operators overloading

- [GACromosome](#) & [operator=](#) (const [GACromosome](#) &c)  
*Overload of assignment operator.*
- bool [operator<](#) (const [GACromosome](#) &c) const  
*Overload of less-than operator.*

### Evolve functions

- void [mutation](#) ()  
*Change one gene randomly.*
- void [add\\_breeding](#) ()  
*Add one breeding.*
- void [use\\_breeding](#) ()  
*Use one breeding.*

### Friends

- std::istream & [operator>>](#) (std::istream &in, [GACromosome](#) &c)  
*<< overloading.*
- std::ostream & [operator<<](#) (std::ostream &in, [GACromosome](#) &c)  
*>> overloading.*

### 10.104.1 Detailed Description

Chromosome class for genetic algorithm.

This is a chromosome class for genetic algorithm, which is an array of genes (int with value of -1, 0 or 1).

Definition at line 46 of file GACHromosome.H.

### 10.104.2 Constructor & Destructor Documentation

#### 10.104.2.1 GACHromosome::GACHromosome ()

Uninitialized constructor. Need to call [resize\(\)](#) later.

Definition at line 43 of file GACHromosome.C.

#### 10.104.2.2 GACHromosome::GACHromosome (const int *N*, const int \* *a*)

Construct from C array.

Definition at line 53 of file GACHromosome.C.

References [init\(\)](#).

#### 10.104.2.3 GACHromosome::GACHromosome (const int *N*)

Construct random chromosome.

Definition at line 47 of file GACHromosome.C.

References [init\(\)](#).

#### 10.104.2.4 GACHromosome::GACHromosome (const [GACHromosome](#) & *c*)

Copy constructor.

Definition at line 59 of file GACHromosome.C.

References [genes](#), [init\(\)](#), and [size](#).

#### 10.104.2.5 GACHromosome::~~GACHromosome ()

Destructor.

Definition at line 86 of file GACHromosome.C.

### 10.104.3 Member Function Documentation

#### 10.104.3.1 void GChromosome::add\_breeding ()

Add one breeding.

Definition at line 161 of file GChromosome.C.

Referenced by GAPopulation::selection().

#### 10.104.3.2 int GChromosome::get\_breedings () const

Return the breedings.

Definition at line 136 of file GChromosome.C.

Referenced by GAPopulation::crossover().

#### 10.104.3.3 float GChromosome::get\_fitness () const

Return the fitness.

Definition at line 114 of file GChromosome.C.

Referenced by GAPopulation::compute\_pop\_fitness(), GAPopulation::compute\_sigma(), and GAPopulation::linear\_scaling().

#### 10.104.3.4 int GChromosome::get\_gene (const int i) const

Return the gene #i.

Definition at line 102 of file GChromosome.C.

References ASSERT.

#### 10.104.3.5 float GChromosome::get\_linear\_fitness () const

Return the linear fitness.

Definition at line 125 of file GChromosome.C.

Referenced by GAPopulation::selection().

#### 10.104.3.6 int GChromosome::get\_size () const

Return the size of the chromosome.

Definition at line 91 of file GChromosome.C.



Referenced by GAPopulation::set\_chromosome().

#### 10.104.3.7 void GAChromosome::init (const int *N*)

Initialize randomly.

Definition at line 79 of file GAChromosome.C.

References resize().

#### 10.104.3.8 void GAChromosome::init (const int *N*, const int \* *a*)

Initialize from C array.

Definition at line 73 of file GAChromosome.C.

References resize().

Referenced by GAChromosome(), GAPopulation::init(), and operator=().

#### 10.104.3.9 void GAChromosome::mutation ()

Change one gene randomly.

Definition at line 155 of file GAChromosome.C.

Referenced by GAPopulation::mutate().

#### 10.104.3.10 bool GAChromosome::operator< (const GAChromosome & *c*) const

Overload of less-than operator.

$c1 < c2$  if *c1* as less breedings than *c2*

Definition at line 150 of file GAChromosome.C.

References breedings.

#### 10.104.3.11 GAChromosome & GAChromosome::operator= (const GAChromosome & *c*)

Overload of assignment operator.

Definition at line 141 of file GAChromosome.C.

References breedings, fitness, genes, init(), linear\_fitness, and size.

**10.104.3.12 void GChromosome::resize (const int *N*)**

Change size, do not initialize.

Definition at line 66 of file GChromosome.C.

Referenced by init(), and operator>>().

**10.104.3.13 void GChromosome::set\_breedings (const int *a*)**

Set breedings to *a*.

Definition at line 130 of file GChromosome.C.

References ASSERT.

**10.104.3.14 void GChromosome::set\_fitness (const float *a*)**

Set the fitness to *a*.

Definition at line 108 of file GChromosome.C.

References ASSERT.

**10.104.3.15 void GChromosome::set\_gene (const int *i*, const int *a*)**

Set the gene #*i* to *a*.

Definition at line 96 of file GChromosome.C.

References abs(), and ASSERT.

Referenced by GAPopulation::crossover().

**10.104.3.16 void GChromosome::set\_linear\_fitness (const float *a*)**

Set the linear fitness to *a*.

Definition at line 119 of file GChromosome.C.

References ASSERT.

Referenced by GAPopulation::linear\_scaling().

**10.104.3.17 void GChromosome::use\_breeding ()**

Use one breeding.

Definition at line 166 of file GChromosome.C.

References ASSERT.

Referenced by GAPopulation::crossover().

#### 10.104.4 Friends And Related Function Documentation

##### 10.104.4.1 `std::ostream& operator<< (std::ostream & in, GAChromosome & c) [friend]`

>> overloading.

Definition at line 191 of file GAChromosome.C.

##### 10.104.4.2 `std::istream& operator>> (std::istream & in, GAChromosome & c) [friend]`

<< overloading.

Definition at line 172 of file GAChromosome.C.

The documentation for this class was generated from the following files:

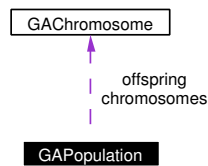
- [GAChromosome.H](#)
- [GAChromosome.C](#)

## 10.105 GAPopulation Class Reference

Population class for genetic algorithm.

```
#include <GAPopulation.H>
```

Collaboration diagram for GAPopulation:



### Public Member Functions

#### Constructor and destructor

- [GAPopulation](#) ()  
*Uninitialized constructor. Need to call [resize\(\)](#) later.*
- [GAPopulation](#) (const int N, const int a)  
*Construct a size N-population of random chromosomes of size a.*
- void [resize](#) (const int N, const int a)  
*Change sizes, do not initialize.*
- void [init](#) (const int N, const int a)  
*Initialize with N random chromosomes of size a.*
- [~GAPopulation](#) ()  
*Destructor.*

#### Access functions

- void [set\\_chromosome](#) (const int i, const [GACHromosome](#) &c)  
*Set the chromosome #i to c.*
- [GACHromosome](#) [get\\_chromosome](#) (const int i) const  
*Return the chromosome #i.*
- float [get\\_mean\\_fitness](#) () const  
*Return the mean fitness.*

- float `get_sigma ()` const  
*Return the standard deviation.*

### Evolution functions

- void `update ()`  
*The offspring become the new generation.*
- void `compute_pop_fitness ()`  
*Compute the mean and total fitness of the chromosomes.*
- void `compute_sigma ()`  
*Compute the standard deviation of the chromosomes' fitness.*
- void `linear_scaling ()`  
*Apply a linear scaling to the chromosomes' fitness.*
- void `selection ()`  
*Select chromosomes for breeding.*
- void `crossover ()`  
*Breed chromosomes to produce the offspring.*
- void `mutate ()`  
*Mutate the offspring with probability 1/size.*

### Friends

- `std::istream & operator>> (std::istream &in, GAPopulation &pop)`
- `std::ostream & operator<< (std::ostream &out, GAPopulation &pop)`

#### 10.105.1 Detailed Description

Population class for genetic algorithm.

This is a population class for genetic algorithm. It's a size N array of chromosomes, and their potential offspring and comes with various method to handle them.

Definition at line 48 of file GAPopulation.H.

## 10.105.2 Constructor & Destructor Documentation

### 10.105.2.1 `GAPopulation::GAPopulation ()`

Uninitialized constructor. Need to call [resize\(\)](#) later.

Definition at line 49 of file `GAPopulation.C`.

### 10.105.2.2 `GAPopulation::GAPopulation (const int N, const int a)`

Construct a size *N*-population of random chromosomes of size *a*.

Definition at line 54 of file `GAPopulation.C`.

References `init()`.

### 10.105.2.3 `GAPopulation::~~GAPopulation ()`

Destructor.

Definition at line 84 of file `GAPopulation.C`.

## 10.105.3 Member Function Documentation

### 10.105.3.1 `void GAPopulation::compute_pop_fitness ()`

Compute the mean and total fitness of the chromosomes.

Definition at line 120 of file `GAPopulation.C`.

References `GChromosome::get_fitness()`.

### 10.105.3.2 `void GAPopulation::compute_sigma ()`

Compute the standard deviation of the chromosomes' fitness.

Definition at line 128 of file `GAPopulation.C`.

References `GChromosome::get_fitness()`, and `sqrt()`.

### 10.105.3.3 `void GAPopulation::crossover ()`

Breed chromosomes to produce the offspring.

Definition at line 179 of file `GAPopulation.C`.

References `GACromosome::get_breedings()`, `GACromosome::set_gene()`, and `GACromosome::use_breeding()`.

#### 10.105.3.4 `GACromosome` `GAPopulation::get_chromosome (const int i) const`

Return the chromosome #i.

Definition at line 99 of file `GAPopulation.C`.

#### 10.105.3.5 `float` `GAPopulation::get_mean_fitness () const`

Return the mean fitness.

Definition at line 104 of file `GAPopulation.C`.

#### 10.105.3.6 `float` `GAPopulation::get_sigma () const`

Return the standard deviation.

Definition at line 109 of file `GAPopulation.C`.

#### 10.105.3.7 `void` `GAPopulation::init (const int N, const int a)`

Initialize with N random chromosomes of size a.

Definition at line 74 of file `GAPopulation.C`.

References `GACromosome::init()`, and `resize()`.

Referenced by `GAPopulation()`.

#### 10.105.3.8 `void` `GAPopulation::linear_scaling ()`

Apply a linear scaling to the chromosomes' fitness.

Definition at line 140 of file `GAPopulation.C`.

References `GACromosome::get_fitness()`, and `GACromosome::set_linear_fitness()`.

#### 10.105.3.9 `void` `GAPopulation::mutate ()`

Mutate the offspring with probability 1/size.

Definition at line 214 of file `GAPopulation.C`.

References `GACromosome::mutation()`.

**10.105.3.10 void GAPopulation::resize (const int *N*, const int *a*)**

Change sizes, do not initialize.

Definition at line 61 of file GAPopulation.C.

Referenced by init().

**10.105.3.11 void GAPopulation::selection ()**

Select chromosomes for breeding.

Definition at line 161 of file GAPopulation.C.

References GACHromosome::add\_breeding(), and GACHromosome::get\_linear\_fitness().

**10.105.3.12 void GAPopulation::set\_chromosome (const int *i*, const [GACHromosome](#) & *c*)**

Set the chromosome #*i* to *c*.

Definition at line 93 of file GAPopulation.C.

References ASSERT, and GACHromosome::get\_size().

**10.105.3.13 void GAPopulation::update ()**

The offspring become the new generation.

Definition at line 114 of file GAPopulation.C.

The documentation for this class was generated from the following files:

- [GAPopulation.H](#)
- [GAPopulation.C](#)

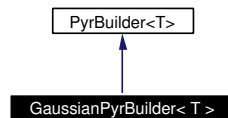


## 10.106 GaussianPyrBuilder< T > Class Template Reference

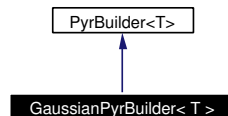
Builds pyramids based on Gaussian filters.

```
#include <PyrBuilder.H>
```

Inheritance diagram for GaussianPyrBuilder< T >:



Collaboration diagram for GaussianPyrBuilder< T >:



### Public Member Functions

- [GaussianPyrBuilder](#) (const int filter\_size)  
*Constructor.*
- virtual [GaussianPyrBuilder< T > \\* clone \(\)](#) const  
*Cloning constructor:.*
- virtual [ImageSet< T > build](#) (const [Image< T >](#) &img, const int depth)  
*Builds a Gaussian pyramid based on img.*

### 10.106.1 Detailed Description

```
template<class T> class GaussianPyrBuilder< T >
```

Builds pyramids based on Gaussian filters.

Definition at line 94 of file `PyrBuilder.H`.

## 10.106.2 Constructor & Destructor Documentation

### 10.106.2.1 `template<class T> GaussianPyrBuilder< T >::GaussianPyrBuilder(const int filter_size)`

Constructor.

Definition at line 71 of file `PyrBuilder.C`.

## 10.106.3 Member Function Documentation

### 10.106.3.1 `template<class T> ImageSet< T > GaussianPyrBuilder< T >::build(const Image< T > & img, const int depth)` [virtual]

Builds a Gaussian pyramid based on *img*.

Implements [PyrBuilder](#)< T >.

Definition at line 76 of file `PyrBuilder.C`.

References `buildPyrGaussian()`.

Referenced by `PyrFoveator::foveate()`.

### 10.106.3.2 `template<class T> GaussianPyrBuilder< T > * GaussianPyrBuilder< T >::clone() const` [inline, virtual]

Cloning constructor:.

Implements [PyrBuilder](#)< T >.

Definition at line 80 of file `PyrBuilder.C`.

The documentation for this class was generated from the following files:

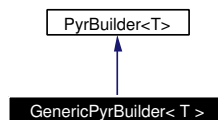
- [PyrBuilder.H](#)
- [PyrBuilder.C](#)

## 10.107 GenericPyrBuilder< T > Class Template Reference

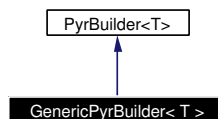
Builds pyramids of a type specified by a PyramidType.

```
#include <PyrBuilder.H>
```

Inheritance diagram for GenericPyrBuilder< T >:



Collaboration diagram for GenericPyrBuilder< T >:



### Public Member Functions

- `GenericPyrBuilder` (const `PyramidType` typ, const float gabor\_theta=0.0F, const float intens=10.0F)  
*Build with depth & type, but no input image yet.*
- virtual `GenericPyrBuilder< T > * clone ()` const  
*Cloning constructor:.*
- virtual `ImageSet< T > build` (const `Image< T > &image`, const int depth)  
*Create a pyramid from an input image.*

### 10.107.1 Detailed Description

```
template<class T> class GenericPyrBuilder< T >
```

Builds pyramids of a type specified by a PyramidType.

This is a chameleon class; it can build any kind of pyramid (Gaussian, Laplacian, Gabor, QuickLocalAvg, QuickLocalMax) depending on the PyramidType passed to its constructor.

Definition at line 219 of file PyrBuilder.H.

## 10.107.2 Constructor & Destructor Documentation

**10.107.2.1** `template<class T> GenericPyrBuilder< T >::GenericPyrBuilder (const PyramidType typ, const float gabor_theta = 0.0F, const float intens = 10.0F)`

Build with depth & type, but no input image yet.

*gabor\_theta* and *intens* are used for Gabor pyramids.

Definition at line 198 of file PyrBuilder.C.

## 10.107.3 Member Function Documentation

**10.107.3.1** `template<class T> ImageSet< T > GenericPyrBuilder< T >::build (const Image< T > & image, const int depth) [virtual]`

Create a pyramid from an input image.

The type of pyramid that is built is determined by the PyramidType that was given to the constructor.

Implements [PyrBuilder](#)< T >.

Definition at line 206 of file PyrBuilder.C.

References [buildPyrGeneric\(\)](#), and [image](#).

**10.107.3.2** `template<class T> GenericPyrBuilder< T > * GenericPyrBuilder< T >::clone () const [virtual]`

Cloning constructor:.

Implements [PyrBuilder](#)< T >.

Definition at line 213 of file PyrBuilder.C.

The documentation for this class was generated from the following files:

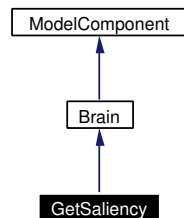
- [PyrBuilder.H](#)
- [PyrBuilder.C](#)

## 10.108 GetSaliency Class Reference

This class allows for easy access to the list of salient points in an image.

```
#include <getSaliency.H>
```

Inheritance diagram for GetSaliency:



Collaboration diagram for GetSaliency:



### Public Member Functions

- [GetSaliency](#) ([ModelManager](#) &mgr)  
*Constructor.*
- [~GetSaliency](#) ()  
*Destructor.*
- const int [compute](#) (const [Image](#)< [PixRGB](#)< [byte](#) > > &img, const [Image](#)< [byte](#) > &targets, const double max\_time)  
*Compute the table of salient spots.*
- const [Image](#)< float > & [getSalmap](#) ()  
*Returns the initial saliency map **prior** to the evolution.*
- const [std::vector](#)< [Point2D](#) > & [getCoords](#) ()  
*Returns the coordinates of the salient spots.*

- const `std::vector< double >` & `getTimes ()`

*Returns the evolution times.*

## Protected Attributes

- `Image< float >` `salmap`
- `std::vector< Point2D >` `coords`
- `std::vector< double >` `times`

### 10.108.1 Detailed Description

This class allows for easy access to the list of salient points in an image.

It uses the `Brain` class in order to create a list of the coordinates and evolution times for the salient spots of an image.

Definition at line 47 of file `getSaliency.H`.

### 10.108.2 Constructor & Destructor Documentation

#### 10.108.2.1 `GetSaliency::GetSaliency (ModelManager & mgr)`

Constructor.

Definition at line 52 of file `getSaliency.C`.

References `Brain::addModule()`.

#### 10.108.2.2 `GetSaliency::~~GetSaliency ()`

Destructor.

Definition at line 80 of file `getSaliency.C`.

### 10.108.3 Member Function Documentation

#### 10.108.3.1 `const int GetSaliency::compute (const Image< PixRGB< byte > > & img, const Image< byte > & targets, const double max_time)`

Compute the table of salient spots.

Required parameters:

**Parameters:**

**inp** input color image

**targets** binary map of the same size as the input, in which targets for the focus of attention are 255 and the rest is zero. Pass an uninitialized image if you have no targets to look for.

**Returns:**

The number of salient spots found within max\_time seconds.

Definition at line 85 of file getSaliency.C.

References Brain::evolve(), Brain::getVC(), Point2D::i, and Brain::input().

**10.108.3.2 const std::vector< Point2D > & GetSaliency::getCoords ()**

Returns the coordinates of the salient spots.

**Returns:**

A vector of Point2D objects - the x and y coordinates of the salient spots.

Definition at line 119 of file getSaliency.C.

**10.108.3.3 const Image< float > & GetSaliency::getSalmap ()**

Returns the initial saliency map **prior** to the evolution.

Should be called after compute.

**Returns:**

The saliency map prior to the evolutions, normalized between 0.0 and 1.0 - the size is the same as the input image.

Definition at line 115 of file getSaliency.C.

**10.108.3.4 const std::vector< double > & GetSaliency::getTimes ()**

Returns the evolution times.

**Returns:**

A vector of double numbers that contains the evolution times for the salient spots.

Definition at line 123 of file getSaliency.C.

The documentation for this class was generated from the following files:

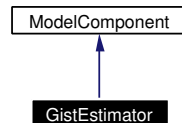
- [getSaliency.H](#)
- [getSaliency.C](#)



## 10.109 GistEstimator Class Reference

```
#include <GistEstimator.H>
```

Inheritance diagram for GistEstimator:



Collaboration diagram for GistEstimator:

### Public Member Functions

- [GistEstimator](#) ([ModelManager](#) &mgr, const [std::string](#) &descrName="Gist Estimator", const [std::string](#) &tagName="gistestimator")

*Constructor.*

- virtual [~GistEstimator](#) ()

*Destructor.*

- virtual void [exportOptions](#) (const int whatToExport, const bool recurse=true)

*Export available options.*

- void [setVC](#) (const [SharedPtr](#)< [VisualCortex](#) > &inVC)

*set the Visual Cortex if it was not given at construction*

- void [compute](#) ()

*compute gist features*

- double \* [getGist](#) ()

### 10.109.1 Detailed Description

implementation of the gist estimation procedure yet to be published

Definition at line 53 of file GistEstimator.H.

## 10.109.2 Constructor & Destructor Documentation

### 10.109.2.1 GistEstimator::GistEstimator ([ModelManager](#) & mgr, const [std::string](#) & descrName = "Gist Estimator", const [std::string](#) & tagName = "gistestimator")

Constructor.

**Parameters:**

**brain** this is the hook for the class to obtain its information on channels, submaps etc.

Definition at line 79 of file GistEstimator.C.

### 10.109.2.2 GistEstimator::~GistEstimator () [virtual]

Destructor.

Definition at line 88 of file GistEstimator.C.

## 10.109.3 Member Function Documentation

### 10.109.3.1 void GistEstimator::compute ()

compute gist features

Definition at line 111 of file GistEstimator.C.

### 10.109.3.2 void GistEstimator::exportOptions (const int *whatToExport*, const bool *recurse* = true) [virtual]

Export available options.

Reimplemented from [ModelComponent](#).

Definition at line 95 of file GistEstimator.C.

### 10.109.3.3 void GistEstimator::setVC (const [SharedPtr](#)< [VisualCortex](#) > & inVC)

set the Visual Cortex if it was not given at construction

Definition at line 100 of file GistEstimator.C.

The documentation for this class was generated from the following files:

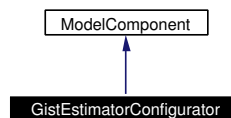
- [GistEstimator.H](#)
- [GistEstimator.C](#)

## 10.110 GistEstimatorConfigurator Class Reference

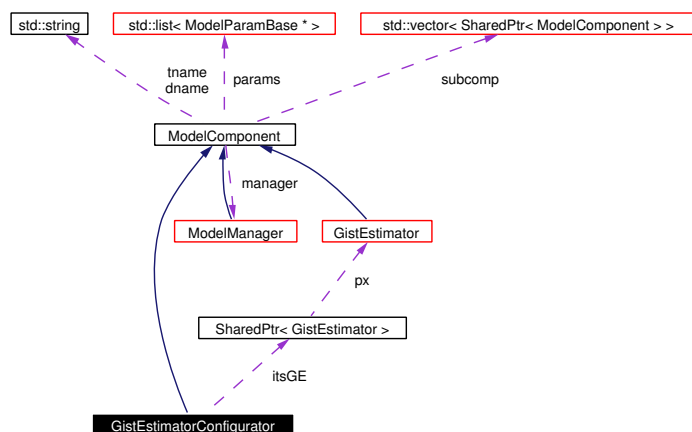
[GistEstimator](#) configurator.

```
#include <GistEstimatorConfigurator.H>
```

Inheritance diagram for GistEstimatorConfigurator:



Collaboration diagram for GistEstimatorConfigurator:



### Public Member Functions

- [GistEstimatorConfigurator](#) ([ModelManager](#) &mgr, const [std::string](#) &descr-Name="Gist Estimator Configurator", const [std::string](#) &tagName="Gist-EstimatorConfigurator")

*Constructor.*

- virtual [~GistEstimatorConfigurator](#) ()

*destructor*

- virtual void [exportOptions](#) (const int whatToExport, const bool recurse=true)

*Export some of our [ModelParam](#) parameters as command-line-options.*

- `SharedPtr< GistEstimator > getGE () const`

*Get the chosen [GistEstimator](#).*

## Protected Member Functions

- virtual bool `setModelParamString (const std::string &name, const std::string &value, const bool recurse=false, const bool warn=true)`

*Intercept people changing our [ModelParam](#).*

## Protected Attributes

- `ModelParam< std::string > itsGEGtype`

*type of Gist Estimator*

### 10.110.1 Detailed Description

[GistEstimator](#) configurator.

This will export the `-ge-type=XX` command-line option and will instantiate a [Gist-Estimator](#) of the desired type as the option gets assigned a value. As this happens, new options may become available in the command-line. To see them, use `-help` AFTER you have chosen the type to use. The current [InferoTemporal](#) may be retrieved using `getGE()`.

Definition at line 49 of file `GistEstimatorConfigurator.H`.

### 10.110.2 Constructor & Destructor Documentation

- 10.110.2.1 `GistEstimatorConfigurator::GistEstimatorConfigurator (ModelManager & mgr, const std::string & descrName = "Gist Estimator Configurator", const std::string & tagName = "GistEstimatorConfigurator")`**

Constructor.

Definition at line 44 of file `GistEstimatorConfigurator.C`.

References `OPTEXP_NONE`, and `SharedPtr< GistEstimator >::reset()`.

### 10.110.2.2 `GistEstimatorConfigurator::~~GistEstimatorConfigurator ()` [virtual]

destructor

Definition at line 56 of file `GistEstimatorConfigurator.C`.

## 10.110.3 Member Function Documentation

### 10.110.3.1 `void GistEstimatorConfigurator::exportOptions (const int whatToExport, const bool recurse = true) [virtual]`

Export some of our [ModelParam](#) parameters as command-line-options.

See the base function in [ModelComponent.H](#) for info. Actually, here we just keep a copy of `whatToExport` and we will call `exportOptions()` on our [GistEstimator](#) once we know its type and instantiate it.

Reimplemented from [ModelComponent](#).

Definition at line 60 of file `GistEstimatorConfigurator.C`.

References `ModelComponent::exportOptions()`, `itsGType`, `OPTEXP_CORE`, and `ModelManager::requestOption()`.

### 10.110.3.2 `SharedPtr< GistEstimator > GistEstimatorConfigurator::getGE () const`

Get the chosen [GistEstimator](#).

You should call this during `start()` of the [ModelComponent](#) that needs the [Gist-Estimator](#). CAUTION: this may be a `SharedPtr(NULL)` if no controller has been chosen.

Definition at line 76 of file `GistEstimatorConfigurator.C`.

### 10.110.3.3 `bool GistEstimatorConfigurator::setModelParamString (const std::string & name, const std::string & value, const bool recurse = false, const bool warn = true) [protected, virtual]`

Intercept people changing our [ModelParam](#).

See [ModelComponent.H](#); as parsing the command-line or reading a config file sets our name, we'll also here instantiate a controller of the proper type (and export its options)

Reimplemented from [ModelComponent](#).

Definition at line 81 of file `GistEstimatorConfigurator.C`.

References `ModelComponent::addSubComponent()`, `SharedPtr< GistEstimator >::get()`, `itsGType`, `ModelComponent::removeSubComponent()`, `SharedPtr< GistEstimator >::reset()`, and `ModelComponent::setModelParamString()`.

#### 10.110.4 Member Data Documentation

##### 10.110.4.1 `ModelParam<std::string> GistEstimatorConfigurator::itsGType` [protected]

type of Gist Estimator

Definition at line 77 of file `GistEstimatorConfigurator.H`.

Referenced by `exportOptions()`, and `setModelParamString()`.

The documentation for this class was generated from the following files:

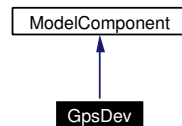
- [GistEstimatorConfigurator.H](#)
- [GistEstimatorConfigurator.C](#)

## 10.111 GpsDev Class Reference

Interface with a GPS unit.

```
#include <GPS.H>
```

Inheritance diagram for GpsDev:



Collaboration diagram for GpsDev:

### Public Member Functions

- [GpsDev](#) ([ModelManager](#) &mgr, const [std::string](#) &descrName="GPS Driver", const [std::string](#) &tagName="GPSdriver")
- virtual [~GpsDev](#) ()  
*destructor*
- int [GPSCheck](#) ()
- [time\\_t](#) [GPSgetTime](#) ()
- void [GPSgetWayPoints](#) ([std::vector](#)< [WayPoint](#) > &WayPointList)
- void [GPSgetLocation](#) ([GPSLocation](#) \*gl)
- void [GPSPvtON](#) ()
- void [GPSgetPVTData](#) ([GPSPvtData](#) \*gd)
- void [GPSPvtOFF](#) ()

### Protected Member Functions

- void [GPSsend](#) ([byte](#) \*message, int bytes)
- int [GPSrecv](#) ()
- short [GPStoshort](#) ([byte](#) \*p)
- float [GPStofloat](#) ([byte](#) \*p)
- long int [GPStolong](#) ([byte](#) \*p)
- double [GPStodouble](#) ([byte](#) \*p)
- double [GPSint2deg](#) (long int n)



### 10.111.1 Detailed Description

Interface with a GPS unit.

Definition at line 46 of file GPS.H.

### 10.111.2 Constructor & Destructor Documentation

#### 10.111.2.1 GpsDev::GpsDev ([ModelManager](#) & *mgr*, const [std::string](#) & *descrName* = "GPS Driver", const [std::string](#) & *tagName* = "GPSdriver")

Default constructor Caution: The settings work properly for the garmin "etrex" model, But will probably need to be tweaked for other GPS models.

**Parameters:**

*dev\_name* is the name of the serial port to which the the device is connected (e.g. /dev/ttyS0, /dev/ttyS1 ..etc)

Definition at line 29 of file GPS.C.

References [ModelComponent::addSubComponent\(\)](#).

#### 10.111.2.2 GpsDev::~GpsDev () [virtual]

destructor

Definition at line 42 of file GPS.C.

### 10.111.3 Member Function Documentation

#### 10.111.3.1 int GpsDev::GPSCheck ()

This checks if the GPS is functioning properly. So it should called before making other calls to within this class.

**Returns:**

0 indicates device failure/misconfiguration, 1 indicates a valid device was found.

Definition at line 81 of file GPS.C.

References [GPSsend\(\)](#).

#### 10.111.3.2 void GpsDev::GPSgetLocation (GPSLocation \* *gl*)

Gives the current longitudinal/latitudinal position

**Parameters:**

*gl* is a place holder for the location information

Definition at line 276 of file GPS.C.

References GPSrecv(), GPSsend(), and GPStodouble().

**10.111.3.3 void GpsDev::GPSgetPVTData (GPSPvtData \* *gd*)**

Get the PVT datum. Each datum contains a wealth of information - current latitude, longitude, velocity in N and E directions, altitude etc.

**Parameters:**

*gd* is a pointer to hold the data, *gd* is \*NOT\* preserved during successive calls

Definition at line 313 of file GPS.C.

References GPSrecv().

**10.111.3.4 time\_t GpsDev::GPSgetTime ()**

The GPS system gives time with an accuracy of +/- 340 ns

**Returns:**

UTC time obtained from the GPS system with an error of a few microseconds

Definition at line 198 of file GPS.C.

References GPSrecv(), GPSsend(), and GPStoshort().

**10.111.3.5 void GpsDev::GPSgetWayPoints (std::vector< WayPoint > & WayPointList)**

Returns a list of longitudes/latitudes obtained from the GPS system.

**Parameters:**

*WayPointList* is the list of WayPoints in which the the latitude and longitude are stired.

Definition at line 246 of file GPS.C.

References GPSint2deg(), GPSrecv(), GPSsend(), and GPStolong().

**10.111.3.6 double GpsDev::GPSint2deg (long int *n*) [protected]**

converts a long int degree value into a double

**Parameters:**

*is* the degree in long int

**Returns:**

is the double value in radians

Definition at line 232 of file GPS.C.

Referenced by GPSgetWayPoints().

**10.111.3.7 void GpsDev::GPSPvtOFF ()**

End the PVT mode and switch back to garmin mode Call this after you are finished with getting PVT data

Definition at line 305 of file GPS.C.

References GPSsend().

**10.111.3.8 void GpsDev::GPSPvtON ()**

Initialize the receiver into the PVT(Position, Velocity, Time) mode. Call this function before calling GPSgetPVTData !

Definition at line 297 of file GPS.C.

References GPSsend().

**10.111.3.9 int GpsDev::GPSrecv () [protected]**

GPSrecv reads from the GPS device into gGPSMessage buffer

**Returns:**

number of bytes read

Definition at line 117 of file GPS.C.

Referenced by GPSgetLocation(), GPSgetPVTData(), GPSgetTime(), and GPSgetWayPoints().

**10.111.3.10 void GpsDev::GPSsend (byte \* message, int bytes) [protected]**

GPSsend is the analog of the socket send and writes to the device the contents of gGPSMessageBuffer

**Parameters:**

*message* is the

*bytes* is the number of bytes to write

Definition at line 46 of file GPS.C.

Referenced by GPSCheck(), GPSgetLocation(), GPSgetTime(), GPSgetWayPoints(), GPSPvtOFF(), and GPSPvtON().

#### 10.111.3.11 double GpsDev::GPStodouble (byte \* p) [protected]

convert 8 adjoining chars into a single 64 bit long int value

**Parameters:**

*p* is the address if the first of the 8 characters

**Returns:**

is the double value

Definition at line 175 of file GPS.C.

Referenced by GPSgetLocation().

#### 10.111.3.12 float GpsDev::GPStofloat (byte \* p) [protected]

convert 4 adjoining chars into a single 32 bit floating point value

**Parameters:**

*p* is the address if the first of the 4 characters

**Returns:**

is the floating point value

Definition at line 165 of file GPS.C.

#### 10.111.3.13 long int GpsDev::GPStolong (byte \* p) [protected]

convert 4 adjoining chars into a single 32 bit long int value

**Parameters:**

*p* is the address if the first of the 4 characters

**Returns:**

is the long int value

Definition at line 184 of file GPS.C.

Referenced by GPSgetWayPoints().

**10.111.3.14** short GpsDev::GPStoshort ([byte](#) \* *p*) [protected]

convert 2 adjoining chars into a single short int

**Parameters:**

*p* is the address of the first of the 2 characters

**Returns:**

is the value of the short

Definition at line 153 of file GPS.C.

Referenced by GPSgetTime().

The documentation for this class was generated from the following files:

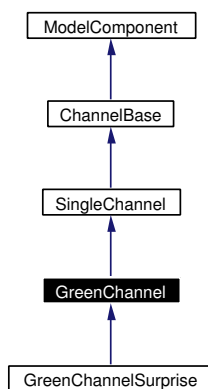
- [GPS.H](#)
- [GPS.C](#)

## 10.112 GreenChannel Class Reference

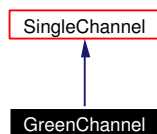
A green color channel.

```
#include <Channels.H>
```

Inheritance diagram for GreenChannel:



Collaboration diagram for GreenChannel:



### Public Member Functions

- [GreenChannel](#) ([ModelManager](#) &mgr)  
*Construct with standard params.*
- virtual [~GreenChannel](#) ()  
*Destructor.*

#### 10.112.1 Detailed Description

A green color channel.

Definition at line 195 of file Channels.H.

## 10.112.2 Constructor & Destructor Documentation

### 10.112.2.1 GreenChannel::GreenChannel ([ModelManager](#) & *mgr*)

Construct with standard params.

Definition at line 241 of file Channels.C.

References [G](#), [makeShared\(\)](#), and [ModelParam< bool >::setVal\(\)](#).

### 10.112.2.2 GreenChannel::~~[GreenChannel](#) () [virtual]

Destructor.

Definition at line 248 of file Channels.C.

The documentation for this class was generated from the following files:

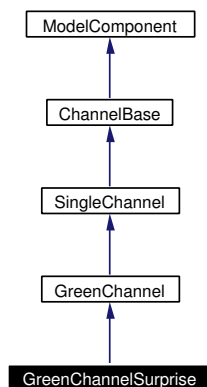
- [Channels.H](#)
- [Channels.C](#)

### 10.113 GreenChannelSurprise Class Reference

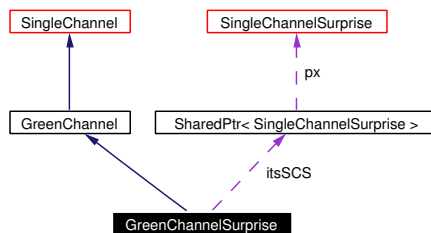
A green color channel.

```
#include <ChannelsSurprise.H>
```

Inheritance diagram for GreenChannelSurprise:



Collaboration diagram for GreenChannelSurprise:



#### Public Member Functions

- [GreenChannelSurprise](#) ([ModelManager](#) &mgr)  
*Construct with standard params.*
- virtual [~GreenChannelSurprise](#) ()  
*Destructor.*
- virtual [Image](#)< float > [computeSubmap](#) (const [uint](#) index) const  
*[computeSubmap\(\)](#) is special and will return surprise*



## Protected Attributes

- [SharedPtr](#)< [SingleChannelSurprise](#) > [itsSCS](#)  
*our surprise computer*

### 10.113.1 Detailed Description

A green color channel.

Definition at line 213 of file ChannelsSurprise.H.

### 10.113.2 Constructor & Destructor Documentation

#### 10.113.2.1 [GreenChannelSurprise::GreenChannelSurprise](#) ([ModelManager](#) & *mgr*)

Construct with standard params.

Definition at line 199 of file ChannelsSurprise.C.

References [ModelComponent::addSubComponent\(\)](#).

#### 10.113.2.2 [GreenChannelSurprise::~~GreenChannelSurprise](#) () [virtual]

Destructor.

Definition at line 205 of file ChannelsSurprise.C.

### 10.113.3 Member Function Documentation

#### 10.113.3.1 [Image](#)< float > [GreenChannelSurprise::computeSubmap](#) (const [uint](#) *index*) const [virtual]

[computeSubmap\(\)](#) is special and will return surprise

Reimplemented from [SingleChannel](#).

Definition at line 208 of file ChannelsSurprise.C.

References [uint](#).

### 10.113.4 Member Data Documentation

#### 10.113.4.1 [SharedPtr<SingleChannelSurprise> GreenChannelSurprise::its-SCS](#) [protected]

our surprise computer

Definition at line 226 of file ChannelsSurprise.H.

The documentation for this class was generated from the following files:

- [ChannelsSurprise.H](#)
- ChannelsSurprise.C

## 10.114 Heading Class Reference

A simple class to represent 3D heading information.

```
#include <HMR3300.H>
```

### Public Member Functions

- [Heading \(\)](#)  
*Constructor.*
- [Heading \(const float h, const float p, const float r\)](#)  
*Initialized constructor.*
- [~Heading \(\)](#)  
*Destructor.*
- float [getHead \(\)](#) const  
*Access function.*
- float [getPitch \(\)](#) const  
*Access function.*
- float [getRoll \(\)](#) const  
*Access function.*
- void [setHead \(const float h\)](#)  
*Access function.*
- void [setPitch \(const float p\)](#)  
*Access function.*
- void [setRoll \(const float r\)](#)  
*Access function.*

### 10.114.1 Detailed Description

A simple class to represent 3D heading information.

Definition at line 45 of file HMR3300.H.

## 10.114.2 Constructor & Destructor Documentation

### 10.114.2.1 Heading::Heading ()

Constructor.

Definition at line 46 of file HMR3300.C.

### 10.114.2.2 Heading::Heading (const float *h*, const float *p*, const float *r*)

Initialized constructor.

Definition at line 50 of file HMR3300.C.

### 10.114.2.3 Heading::~~Heading ()

Destructor.

Definition at line 54 of file HMR3300.C.

## 10.114.3 Member Function Documentation

### 10.114.3.1 float Heading::getHead () const

Access function.

Definition at line 57 of file HMR3300.C.

Referenced by BeoSub::getHeading().

### 10.114.3.2 float Heading::getPitch () const

Access function.

Definition at line 59 of file HMR3300.C.

### 10.114.3.3 float Heading::getRoll () const

Access function.

Definition at line 61 of file HMR3300.C.

### 10.114.3.4 void Heading::setHead (const float *h*)

Access function.

Definition at line 64 of file HMR3300.C.

Referenced by HMR3300::get().

#### 10.114.3.5 void Heading::setPitch (const float *p*)

Access function.

Definition at line 66 of file HMR3300.C.

Referenced by HMR3300::get().

#### 10.114.3.6 void Heading::setRoll (const float *r*)

Access function.

Definition at line 68 of file HMR3300.C.

Referenced by HMR3300::get().

The documentation for this class was generated from the following files:

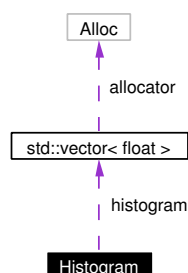
- [HMR3300.H](#)
- [HMR3300.C](#)

## 10.115 Histogram Class Reference

Histogram.

```
#include <Histogram.H>
```

Collaboration diagram for Histogram:



### Public Member Functions

- [Histogram](#) ()  
*Constructor.*
- [Histogram](#) (int size)  
*Constructor.*
- [~Histogram](#) ()  
*Destructor.*
- float [getValue](#) (int index)  
*get a value from the histogram*
- void [addValue](#) (int index, float value)  
*add a value to the histogram*
- void [smooth](#) ()  
*smooth the histogram*
- float [findMax](#) ()  
*Retrieve the peak of the histogram.*

### 10.115.1 Detailed Description

Histogram.

Definition at line 42 of file Histogram.H.

### 10.115.2 Constructor & Destructor Documentation

#### 10.115.2.1 Histogram::Histogram () [inline]

Constructor.

Definition at line 46 of file Histogram.H.

#### 10.115.2.2 Histogram::Histogram (int size)

Constructor.

Definition at line 46 of file Histogram.C.

#### 10.115.2.3 Histogram::~Histogram () [inline]

Destructor.

Definition at line 54 of file Histogram.H.

### 10.115.3 Member Function Documentation

#### 10.115.3.1 void Histogram::addValue (int *index*, float *value*)

add a value to the histogram

Definition at line 52 of file Histogram.C.

References uint.

Referenced by VisualObject::createKeys().

#### 10.115.3.2 float Histogram::findMax ()

Retrieve the peak of the histogram.

Definition at line 88 of file Histogram.C.

References uint.

Referenced by VisualObject::createKeys().

**10.115.3.3 float Histogram::getValue (int *index*)**

get a value from the histogram

Definition at line 61 of file Histogram.C.

References uint.

Referenced by VisualObject::createKeys().

**10.115.3.4 void Histogram::smooth ()**

smooth the histogram

Definition at line 70 of file Histogram.C.

References uint.

Referenced by VisualObject::createKeys().

The documentation for this class was generated from the following files:

- [Histogram.H](#)
- [Histogram.C](#)

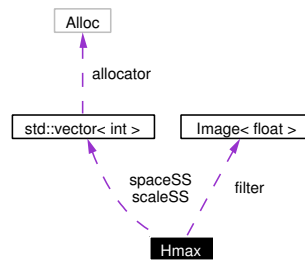


## 10.116 Hmax Class Reference

Object recognition model inspired from Riesenhuber & Poggio's HMAX model.

```
#include <Hmax.H>
```

Collaboration diagram for Hmax:



### Public Member Functions

- [Hmax](#) ()  
*Default constructor. Need to call [init\(\)](#) later.*
- [Hmax](#) (const int nori, const [std::vector](#)< int > &space, const [std::vector](#)< int > &scaless, const int c1spaceol=2, const bool angleflag=true, const float s2t=1.0F, const float s2s=1.0F, const float stdmin=1.75F, const float stdstep=0.5F, const int fsmin=3, const int fsstep=1)  
*Constructor.*
- void [init](#) (const int nori, const [std::vector](#)< int > &space, const [std::vector](#)< int > &scaless, const int c1spaceol=2, const bool angleflag=true, const float s2t=1.0F, const float s2s=1.0F, const float stdmin=1.75F, const float stdstep=0.5F, const int fsmin=3, const int fsstep=1)  
*Initialization.*
- void [freeMem](#) ()  
*Free internal memory.*
- [~Hmax](#) ()  
*Destructor.*
- [vector](#)< [string](#) > [readDir](#) ([string](#) dirName)  
*For utilize the operation : read a directory and pass the list of files.*

- `Image< float > origGetC2 (const Image< float > &imag)`

*This is the code of Max Riesenhuber almost straight out of the box.*

- `Image< float > getC2resp (const Image< byte > &in)`

*Compute C2 responses from an input image.*

- `Image< float > getC2 (const Image< float > &input)`

## Protected Attributes

- `bool initialized`

*keep track of whether we have allocated some memory*

- `int ns`

*number of scales*

- `int no`

*number of orientations*

- `Image< float > ** filter`

*low-level processing filters [ns][no]*

- `int c1SpaceOL`

*spatial overlap in C1 cells*

- `bool angleFlag`

*normalize convolution by image energy if true*

- `float s2Target`

*center of S2 target*

- `float s2Sigma`

*size of S2 target*

- `std::vector< int > spaceSS`

- `std::vector< int > scaleSS`

### 10.116.1 Detailed Description

Object recognition model inspired from Riesenhuber & Poggio's HMAX model.

See Riesenhuber & Poggio, Nature Neuroscience, 1999.

Definition at line 56 of file Hmax.H.

### 10.116.2 Constructor & Destructor Documentation

#### 10.116.2.1 Hmax::Hmax ()

Default constructor. Need to call [init\(\)](#) later.

Definition at line 51 of file Hmax.C.

#### 10.116.2.2 Hmax::Hmax (const int *nori*, const [std::vector](#)< int > & *spaces*, const [std::vector](#)< int > & *scales*, const int *c1spaceol* = 2, const bool *angleflag* = true, const float *s2t* = 1.0F, const float *s2s* = 1.0F, const float *stdmin* = 1.75F, const float *stdstep* = 0.5F, const int *fsmin* = 3, const int *fsstep* = 1)

Constructor.

Definition at line 55 of file Hmax.C.

References [init\(\)](#).

#### 10.116.2.3 Hmax::~~Hmax ()

Destructor.

Definition at line 112 of file Hmax.C.

References [freeMem\(\)](#).

### 10.116.3 Member Function Documentation

#### 10.116.3.1 void Hmax::freeMem ()

Free internal memory.

Definition at line 101 of file Hmax.C.

References [filter](#), and [ns](#).

Referenced by [init\(\)](#), and [~Hmax\(\)](#).

**10.116.3.2** `Image<float> Hmax::getC2resp (const Image< byte > & in)`

Compute C2 responses from an input image.

**10.116.3.3** `void Hmax::init (const int nori, const std::vector< int > & spacess, const std::vector< int > & scaless, const int c1spaceol = 2, const bool angleflag = true, const float s2t = 1.0F, const float s2s = 1.0F, const float stdmin = 1.75F, const float stdstep = 0.5F, const int fsm = 3, const int fsstep = 1)`

Initialization.

Definition at line 67 of file Hmax.C.

References `angleFlag`, `c1SpaceOL`, `filter`, `freeMem()`, `mean()`, `no`, `ns`, `s2Sigma`, `s2Target`, `squared()`, and `sum()`.

Referenced by `Hmax()`.

**10.116.3.4** `Image< float > Hmax::origGetC2 (const Image< float > & imag)`

This is the code of Max Riesenhuber almost straight out of the box.

Definition at line 144 of file Hmax.C.

References `angleFlag`, `c1SpaceOL`, `filter`, `Image< T >::getArrayPtr()`, `Image< float >::getWidth()`, `hmaxActivation()`, `image`, `no`, `ns`, `Image< T >::resize()`, `s2Sigma`, `s2Target`, `safecopy()`, and `sqrt()`.

**10.116.3.5** `vector< string > Hmax::readDir (string dirName)`

For utilize the operation : read a directory and pass the list of files.

Definition at line 116 of file Hmax.C.

**10.116.4 Member Data Documentation****10.116.4.1** `bool Hmax::angleFlag` [protected]

normalize convolution by image energy if true

Definition at line 99 of file Hmax.H.

Referenced by `init()`, and `origGetC2()`.

**10.116.4.2** `int Hmax::c1SpaceOL` [protected]

spatial overlap in C1 cells

Definition at line 98 of file Hmax.H.

Referenced by `init()`, and `origGetC2()`.

**10.116.4.3** `Image<float>*& Hmax::filter` [protected]

low-level processing filters [ns][no]

Definition at line 96 of file Hmax.H.

Referenced by `freeMem()`, `init()`, and `origGetC2()`.

**10.116.4.4** `bool Hmax::initialized` [protected]

keep track of whether we have allocated some memory

Definition at line 93 of file Hmax.H.

**10.116.4.5** `int Hmax::no` [protected]

number of orientations

Definition at line 95 of file Hmax.H.

Referenced by `init()`, and `origGetC2()`.

**10.116.4.6** `int Hmax::ns` [protected]

number of scales

Definition at line 94 of file Hmax.H.

Referenced by `freeMem()`, `init()`, and `origGetC2()`.

**10.116.4.7** `float Hmax::s2Sigma` [protected]

size of S2 target

Definition at line 101 of file Hmax.H.

Referenced by `init()`, and `origGetC2()`.

**10.116.4.8** float [Hmax::s2Target](#) [protected]

center of S2 target

Definition at line 100 of file Hmax.H.

Referenced by [init\(\)](#), and [origGetC2\(\)](#).

The documentation for this class was generated from the following files:

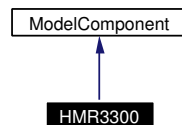
- [Hmax.H](#)
- [Hmax.C](#)

## 10.117 HMR3300 Class Reference

Class for interfacing with a Honeywell 3300 compass.

```
#include <HMR3300.H>
```

Inheritance diagram for HMR3300:



Collaboration diagram for HMR3300:

### Public Member Functions

- [HMR3300](#) ([ModelManager](#) &mgr, const [std::string](#) &descrName="Compass Controller", const [std::string](#) &tagName="CompassControl", const char \*dev="/dev/ttyS0")

*Initialize.*

- [~HMR3300](#) ()

*Destructor.*

- void [get](#) ([Heading](#) &head)

*read the heading, pitch and roll from the compass*

- void [run](#) ()

*The user need not concern himself with this method.*

### Protected Member Functions

- void [start2](#) ()

*get started*

- void [stop1](#) ()

*get stopped*

### 10.117.1 Detailed Description

Class for interfacing with a Honeywell 3300 compass.

Definition at line 76 of file HMR3300.H.

### 10.117.2 Constructor & Destructor Documentation

**10.117.2.1 HMR3300::HMR3300** ([ModelManager](#) & *mgr*, const [std::string](#) & *descrName* = "Compass Controller", const [std::string](#) & *tagName* = "CompassControl", const char \* *dev* = "/dev/ttyS0")

Initialize.

Definition at line 85 of file HMR3300.C.

References [ModelComponent::addSubComponent\(\)](#).

**10.117.2.2 HMR3300::~~HMR3300** ()

Destructor.

Definition at line 110 of file HMR3300.C.

### 10.117.3 Member Function Documentation

**10.117.3.1 void HMR3300::get** ([Heading](#) & *head*)

read the heading, pitch and roll from the compass

Definition at line 140 of file HMR3300.C.

References [Heading::setHead\(\)](#), [Heading::setPitch\(\)](#), and [Heading::setRoll\(\)](#).

**10.117.3.2 void HMR3300::run** ()

The user need not concern himself with this method.

Should have been protected, but is not because of the pthread hack.

Definition at line 114 of file HMR3300.C.

**10.117.3.3 void HMR3300::start2** () [protected, virtual]

get started



Reimplemented from [ModelComponent](#).

Definition at line 99 of file HMR3300.C.

#### 10.117.3.4 void HMR3300::stop1 () [protected, virtual]

get stopped

Reimplemented from [ModelComponent](#).

Definition at line 103 of file HMR3300.C.

The documentation for this class was generated from the following files:

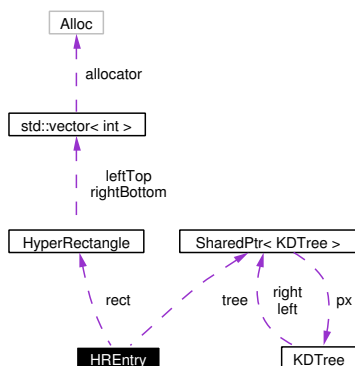
- [HMR3300.H](#)
- [HMR3300.C](#)

## 10.118 HREntry Class Reference

Represents a [HyperRectangle](#) entry in a list.

```
#include <KDTree.H>
```

Collaboration diagram for HREntry:



### Public Member Functions

- [HREntry](#) ()  
*Constructor.*
- [HREntry](#) (const [HyperRectangle](#) &rect, const [SharedPtr< KDTree >](#) &tree, const Keypoint &pivot, double dist)  
*Constructor.*
- const double [getDistance](#) () const  
*Get distance.*
- const [HyperRectangle](#) & [getHR](#) () const  
*Get the [HyperRectangle](#).*
- const Keypoint & [getPivot](#) () const  
*Get the pivot.*
- const [SharedPtr< KDTree >](#) & [getTree](#) () const  
*Get the tree.*
- bool [equals](#) (const [HREntry](#) &obj) const

*Find if this HREntry equals the specified HREntry.*

- `int compareTo (const HREntry &obj) const`  
*Compare this HREntry to the specified HREntry.*

### 10.118.1 Detailed Description

Represents a [HyperRectangle](#) entry in a list.

Definition at line 383 of file KDTree.H.

### 10.118.2 Constructor & Destructor Documentation

#### 10.118.2.1 HREntry::HREntry ()

Constructor.

Definition at line 715 of file KDTree.C.

#### 10.118.2.2 HREntry::HREntry (const [HyperRectangle](#) & rect, const [SharedPtr](#)< [KDTree](#) > & tree, const Keypoint & pivot, double dist)

Constructor.

Definition at line 720 of file KDTree.C.

### 10.118.3 Member Function Documentation

#### 10.118.3.1 int HREntry::compareTo (const [HREntry](#) & obj) const

Compare this HREntry to the specified HREntry.

Definition at line 765 of file KDTree.C.

References [getDistance\(\)](#).

#### 10.118.3.2 bool HREntry::equals (const [HREntry](#) & obj) const

Find if this HREntry equals the specified HREntry.

Definition at line 754 of file KDTree.C.

References [getDistance\(\)](#), and [getPivot\(\)](#).

**10.118.3.3    `const double HREntry::getDistance () const`**

Get distance.

Definition at line 730 of file KDTree.C.

Referenced by `compareTo()`, and `equals()`.

**10.118.3.4    `const HyperRectangle & HREntry::getHR () const`**

Get the [HyperRectangle](#).

Definition at line 736 of file KDTree.C.

**10.118.3.5    `const Keypoint & HREntry::getPivot () const`**

Get the pivot.

Definition at line 742 of file KDTree.C.

Referenced by `equals()`.

**10.118.3.6    `const SharedPtr< KDTree > & HREntry::getTree () const`**

Get the tree.

Definition at line 748 of file KDTree.C.

The documentation for this class was generated from the following files:

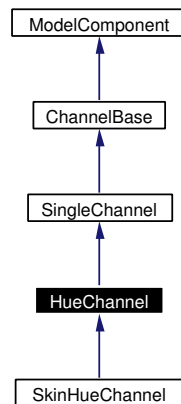
- [KDTree.H](#)
- [KDTree.C](#)

## 10.119 HueChannel Class Reference

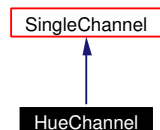
A hue similarity channel.

```
#include <Channels.H>
```

Inheritance diagram for HueChannel:



Collaboration diagram for HueChannel:



### Public Member Functions

- [HueChannel](#) ([ModelManager](#) &mgr, float muR, float muG, float sigR, float sigG, float rho)  
*Constructor with parameters of the bivariate Gaussian.*
- [HueChannel](#) ([ModelManager](#) &mgr, const [PixRGB](#)< [byte](#) > hue, float sig=0.04)  
*Constructor with sample pixel and std.*
- virtual [~HueChannel](#) ()  
*virtual destructor for inheritance*

- `Image< float > combineSubMaps ()`  
*Combine all feature maps into a single output map.*
- `virtual uint numSubmaps () const`  
*always returns 1*
- `virtual Image< float > getSubmap (const uint index)`  
*This is just a caching wrapper around `computeSubmap()`.*
- `virtual const char * getSubmapName (const uint index) const`  
*Always returns "Hue".*

## Protected Member Functions

- `virtual void doInput (const Image< PixRGB< float > > *colimg, const Image< float > *, const double t, const Image< byte > &clipMask)`  
*special input function to compute the hue distance map*

## Protected Attributes

- `float itsMuR`
- `float itsMuG`
- `float itsSigR`
- `float itsSigG`
- `float itsRho`

### 10.119.1 Detailed Description

A hue similarity channel.

Hue is represented as a bivariate Gaussian distribution in the C.I.E. color space, which is R/I vs. G/I.

Definition at line 786 of file Channels.H.

## 10.119.2 Constructor & Destructor Documentation

### 10.119.2.1 HueChannel::HueChannel ([ModelManager](#) & *mgr*, float *muR*, float *muG*, float *sigR*, float *sigG*, float *rho*)

Constructor with parameters of the bivariate Gaussian.

**Parameters:**

- mgr* the [ModelManager](#) to assign this channel to
- muR* the mean of the distribution in the R/I coordinate
- muG* the mean in the G/I coordinate of the C.I.E. color space
- sigR* the std in R/I
- sigG* the std in G/I
- rho* the correlation between R/I and G/I

Definition at line 3033 of file Channels.C.

References [HUE](#), and [makeShared\(\)](#).

### 10.119.2.2 HueChannel::HueChannel ([ModelManager](#) & *mgr*, const [PixRGB](#)<[byte](#) > *hue*, float *sig* = 0.04)

Constructor with sample pixel and std.

Simplified version of the constructor that derives *muR* and *muG* from the pixel value *hue*, and it assumes that *sigR*=*sigG*=*sig*, and *rho* = 0.

Definition at line 3045 of file Channels.C.

References [HUE](#), [makeShared\(\)](#), and [RGBtoCIE\(\)](#).

### 10.119.2.3 HueChannel::~~HueChannel () [virtual]

virtual destructor for inheritance

Definition at line 3060 of file Channels.C.

## 10.119.3 Member Function Documentation

### 10.119.3.1 [Image](#)< float > HueChannel::combineSubMaps () [virtual]

Combine all feature maps into a single output map.

This overrides [SingleChannel::combineSubMaps](#). Instead of doing center-surround and combining across scales, we just return the hue-filtered image at the map resolution.

Reimplemented from [SingleChannel](#).

Definition at line 3076 of file Channels.C.

References [SingleChannel::getImage\(\)](#), [ModelParam< float >::getVal\(\)](#), [ModelParam< MaxNormType >::getVal\(\)](#), [ModelParam< LevelSpec >::getVal\(\)](#), [LevelSpec::mapLevel\(\)](#), [maxNormalize\(\)](#), [maxNormTypeName\(\)](#), and [ModelComponent::tagName\(\)](#).

**10.119.3.2** `void HueChannel::doInput (const Image< PixRGB< float > > * colimg, const Image< float > *, const double t, const Image< byte > & clipMask) [protected, virtual]`

special input function to compute the hue distance map

Reimplemented from [SingleChannel](#).

Definition at line 3064 of file Channels.C.

References [ASSERT](#), [SingleChannel::doInput\(\)](#), and [hueDistance\(\)](#).

**10.119.3.3** `Image< float > HueChannel::getSubmap (const uint index) [virtual]`

This is just a caching wrapper around [computeSubmap\(\)](#).

Definition at line 3095 of file Channels.C.

References [SingleChannel::getOutput\(\)](#), and [uint](#).

**10.119.3.4** `const char * HueChannel::getSubmapName (const uint index) const [virtual]`

Always returns "Hue".

Reimplemented from [SingleChannel](#).

Definition at line 3101 of file Channels.C.

References [uint](#).

**10.119.3.5** `uint HueChannel::numSubmaps () const [virtual]`

always returns 1



Reimplemented from [SingleChannel](#).

Definition at line 3091 of file Channels.C.

References [uint](#).

The documentation for this class was generated from the following files:

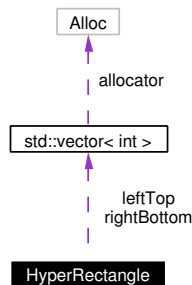
- [Channels.H](#)
- [Channels.C](#)

## 10.120 HyperRectangle Class Reference

Represents a rectangle in n-dimensional space.

```
#include <KDTree.H>
```

Collaboration diagram for HyperRectangle:



### Public Member Functions

- [HyperRectangle \(\)](#)  
*Constructor.*
- [HyperRectangle \(int dim\)](#)  
*Constructor.*
- [HyperRectangle clone \(\)](#) const  
*Clones this HyperRectangle.*
- [HyperRectangle splitAt \(int splitDim, int splitVal\)](#)  
*Splits the rectangle at the specified dimension and value.*
- [bool isIn \(const Keypoint &target\)](#) const  
*Returns true if the target point is within this HR.*
- [bool isInReach \(const Keypoint &target, double distRad\)](#) const  
*Return true if any part of this HR is reachable from target.*
- [double distance \(const Keypoint &target\)](#) const  
*Return the dist from nearest point within the HR to the target point.*

## Static Public Member Functions

- [HyperRectangle createUniverseRectangle](#) (int dim)  
*Creates an all-encompassing rectangle in the dimension specified.*

### 10.120.1 Detailed Description

Represents a rectangle in n-dimensional space.

Definition at line 269 of file KDTree.H.

### 10.120.2 Constructor & Destructor Documentation

#### 10.120.2.1 HyperRectangle::HyperRectangle ()

Constructor.

Definition at line 125 of file KDTree.C.

#### 10.120.2.2 HyperRectangle::HyperRectangle (int *dim*)

Constructor.

Definition at line 130 of file KDTree.C.

### 10.120.3 Member Function Documentation

#### 10.120.3.1 [HyperRectangle](#) HyperRectangle::clone () const

Clones this HyperRectangle.

Definition at line 138 of file KDTree.C.

References leftTop, and rightBottom.

Referenced by splitAt().

#### 10.120.3.2 [HyperRectangle](#) HyperRectangle::createUniverseRectangle (int *dim*) [static]

Creates an all-encompassing rectangle in the dimension specified.

Definition at line 151 of file KDTree.C.

References leftTop, and rightBottom.

Referenced by `KDTree::nearestNeighbor()`, `KDTree::nearestNeighborList()`, and `KDTree::nearestNeighborListBBF()`.

#### 10.120.3.3 `double HyperRectangle::distance (const Keypoint & target) const`

Return the dist from nearest point within the HR to the target point.

Definition at line 207 of file `KDTree.C`.

References `sqrt()`.

Referenced by `isInReach()`.

#### 10.120.3.4 `bool HyperRectangle::isIn (const Keypoint & target) const`

Returns true if the target point is within this HR.

Definition at line 182 of file `KDTree.C`.

#### 10.120.3.5 `bool HyperRectangle::isInReach (const Keypoint & target, double distRad) const`

Return true if any part of this HR is reachable from target.

Must be reachable by no more than 'distRad', false otherwise.

Definition at line 200 of file `KDTree.C`.

References `distance()`.

#### 10.120.3.6 [HyperRectangle](#) `HyperRectangle::splitAt (int splitDim, int splitVal)`

Splits the rectangle at the specified dimension and value.

Definition at line 165 of file `KDTree.C`.

References `clone()`, and `leftTop`.

The documentation for this class was generated from the following files:

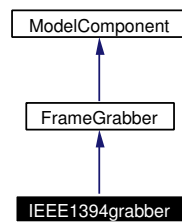
- [KDTree.H](#)
- [KDTree.C](#)

## 10.121 IEEE1394grabber Class Reference

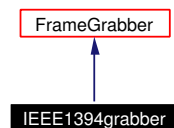
Definition and access functions for FireWire digital camera frame capture.

```
#include <IEEE1394grabber.H>
```

Inheritance diagram for IEEE1394grabber:



Collaboration diagram for IEEE1394grabber:



### Public Member Functions

- [IEEE1394grabber](#) ([ModelManager](#) &mgr, const [std::string](#) &descr-Name="IEEE1394 Frame Grabber Driver", const [std::string](#) &tag-Name="IEEE1394FrameGrabber")
- virtual [~IEEE1394grabber](#) ()  
*Destructor.*
- virtual [Image](#)< [PixRGB](#)< [byte](#) > > [grab](#) ()  
*Grab a color image.*
- virtual [Image](#)< [PixRGB](#)< [byte](#) > > [grabSingle](#) ()  
*Grab a single color image.*

### Protected Member Functions

- virtual void [start1](#) ()

*get started*

- virtual void [stop2\(\)](#)

*get stopped*

### 10.121.1 Detailed Description

Definition and access functions for FireWire digital camera frame capture.

This class provides a trivial interface to FireWire digital cameras (i.e., web cams that send raw frame data over the firewire bus, not DV camcorders that send compressed video streams over the firewire bus). All the low-level setup is done during construction. The user only needs to call the [grab\(\)](#) method to capture an image.

After each grab, the next grab is initiated, and will be ready 33ms later (or one frame later if not NTSC). If you call [grab\(\)](#) again before 33ms have elapsed, it will block until the next frame is available. If you call it too late, you will have missed the latest frame, and [grab\(\)](#) will block until the next frame is acquired.

So a good strategy is to use [grab\(\)](#) to not only grab but also to synchronize your code with the video rate (30 frames/s if NTSC). Typically, then, you would have a main loop that first grabs and then does various processing that is guaranteed to take less than 33ms. You do not need to insert any pause after that processing to obtain a stable framerate; just finish your main loop, and the next call to [grab\(\)](#) (at the next iteration) will block until exactly one frame has passed since it was last called. See how this is done, for example, in `pvisionTCP-master.C` or `test-grab.C`.

Our Unibrain Fire-i rev 1.2 camera supports the following (use `byteswap = false`):

160 x 120: YUV444 (30, 15, 7.5 fps) 320 x 240: YUV422 (30, 15, 7.5, 3.75 fps) 640 x 480: YUV411 (30, 15, 7.5, 3.75 fps) 640 x 480: YUV422 (15, 7.5, 3.75 fps) 640 x 480: RGB24 (15, 7.5, 3.75 fps) 640 x 480: GREY (30, 15, 7.5, 3.75 fps)

To use this, you must have `libdc1394 0.9.0` or later and `libraw1394 0.9.0` or later installed (and you need the `-devel` RPMS installed as well to compile this code). Make sure you also have: `mknod -m 777 /dev/raw1394 c 171 0` `mkdir /dev/video1394` `mknod -m 777 /dev/video1394/0 c 171 16`

Definition at line 96 of file `IEEE1394grabber.H`.

### 10.121.2 Constructor & Destructor Documentation

**10.121.2.1 IEEE1394grabber::IEEE1394grabber** (**ModelManager** & *mgr*, const **std::string** & *descrName* = "IEEE1394 Frame Grabber Driver", const **std::string** & *tagName* = "IEEE1394FrameGrabber")

Constructor Constructor

Definition at line 54 of file IEEE1394grabber.C.

References `ModelParam< bool >::setVal()`, `ModelParam< FrameGrabberMode >::setVal()`, and `ModelParam< int >::setVal()`.

#### 10.121.2.2 IEEE1394grabber::~~IEEE1394grabber () [virtual]

Destructor.

Definition at line 269 of file IEEE1394grabber.C.

### 10.121.3 Member Function Documentation

#### 10.121.3.1 Image< PixRGB< byte > > IEEE1394grabber::grab () [virtual]

Grab a color image.

**Parameters:**

*image* Destination color image. Will be resized to grab width and height as specified in IEEE1394grabber constructor.

Implements [FrameGrabber](#).

Definition at line 273 of file IEEE1394grabber.C.

References `ASSERT`, `FrameGrabber::grab2rgb()`, and `image`.

Referenced by `grabSingle()`.

#### 10.121.3.2 Image< PixRGB< byte > > IEEE1394grabber::grabSingle () [virtual]

Grab a single color image.

This function will work better with single-shot grabs (as opposed to video)

Implements [FrameGrabber](#).

Definition at line 298 of file IEEE1394grabber.C.

References `grab()`.

#### 10.121.3.3 void IEEE1394grabber::start1 () [protected, virtual]

get started

Reimplemented from [ModelComponent](#).

Definition at line 74 of file IEEE1394grabber.C.

References `ModelParam< int >::getVal()`, `ModelParam< float >::getVal()`, `ModelParam< Dims >::getVal()`, `ModelParam< FrameGrabberMode >::getVal()`, `Dims::h()`, `IEEE1394GRABFORMAT`, `IEEE1394GRABSPEED`, and `Dims::w()`.

#### **10.121.3.4 void IEEE1394grabber::stop2 () [protected, virtual]**

get stopped

Reimplemented from [ModelComponent](#).

Definition at line 257 of file IEEE1394grabber.C.

The documentation for this class was generated from the following files:

- [IEEE1394grabber.H](#)
- [IEEE1394grabber.C](#)

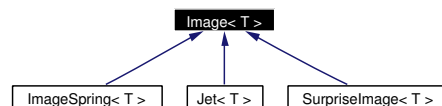


## 10.122 Image< T > Class Template Reference

Generic image template class.

```
#include <Image.H>
```

Inheritance diagram for Image< T >:



### Iterators

There are const and non-const versions of iterators, which are returned by `begin()/end()` and `beginw()/endw()`, respectively. The "w" in `beginw()/endw()` is a mnemonic for "write" or "writeable". Beware that the non-const versions of these functions will invoke the copy-on-write mechanism (see [ArrayHandle](#)), so a deep copy of all the image data will be made if the current image object is not the unique owner of its image data. Thus, `begin()/end()` should always be used unless write access is specifically needed. (It is for this reason that we don't overload the const/non-const functions with the same names (even though this is possible in C++, as done in the STL containers)).

Debugging iterators ([CheckedIterator](#)) will be used if the macro `MEM_DEBUG` is `#define'd`. A checked iterator will check that it is within its proper bounds every time that it is dereferenced with either `operator*()` or `operator → ()`. Note that the existence of the checked iterators means you cannot rely on an image iterator being a raw pointer. Instead, `getArrayPtr()` is available in case you unconditionally need a raw pointer to the image data.

- `typedef T * iterator`  
*Read/write iterator.*
- `typedef const T * const_iterator`  
*Read-only iterator.*
- `const_iterator begin () const`  
*Returns a read-only iterator to the beginning of the image data.*
- `const_iterator end () const`  
*Returns a read-only iterator to one-past-the-end of the image data.*
- `iterator beginw ()`

Returns a read-write iterator to the beginning of the image data.

- `iterator endw ()`

Returns a read-write iterator to one-past-the-end of the image data.

## Public Member Functions

### Constructors, destructors, assignment

- `Image (const T *inarray, int width, int height)`  
Construct from C array.
- `Image (const T *inarray, const Dims &dims)`  
Construct from C array.
- `Image (int width, int height, InitPolicy init)`  
Allocates memory for given size, and optionally zero-clear that memory.
- `Image (const Dims &dims, InitPolicy init)`  
Constructor that only allocates memory for given size.
- `Image ()`  
Construct an empty (0-by-0) image (useful for arrays of Images).
- `Image (const Image< T > &A)`  
Copy constructor.
- `template<class T2> Image (const Image< T2 > &A)`  
Conversion copy constructor.
- `Image< T > & operator= (const Image< T > &A)`  
Assignment operator.
- `Image< T > & operator= (std::vector< std::vector< T > > &A)`  
Conversion assignment operator.
- `Image< T > & operator= (std::vector< std::vector< T > * > &A)`
- `Image< T > & operator= (std::vector< std::vector< T * > > &A)`
- `template<class T2> Image< T > & operator= (const Image< T2 > &A)`  
Conversion assignment operator.
- `~Image ()`  
Destructor.
- `void freeMem ()`

*Free memory and switch to uninitialized state.*

### Memory management functions

- void `swap` (Image< T > &other)  
*Swap the contents of two images.*
- void `attach` (T \*array, const int `width`, const int `height`)  
*Use existing memory.*
- void `detach` ()  
*Detach previously `attach()`'ed image.*
- void `resize` (const `Dims` &dims, const bool clear=false)  
*Free mem and realloc new array (array contents are lost).*
- void `resize` (const int `width`, const int `height`, const bool clear=false)  
*Free mem and realloc new array (array contents are lost).*

### Access functions

- bool `initialized` () const  
*Check whether image is non-empty (i.e., non-zero height and width).*
- int `getSize` () const  
*Get image size (width \* height).*
- int `getWidth` () const  
*Get image width.*
- int `getHeight` () const  
*Get image height.*
- const `Dims` & `getDims` () const  
*Get image width+height in `Dims` struct.*
- template<class C> bool `isSameSize` (const C &other) const  
*Check if \*this is the same size as the other thing.*
- bool `is1D` () const  
*Check if the image is 1D, i.e., width == 1 or height == 1.*
- bool `isFinite` () const  
*Check whether all pixels have finite values.*

- void [getMinMax](#) (T &mini, T &maxi) const  
*Get min and max values.*
- void [getMinMaxAvg](#) (T &mini, T &maxi, T &avg) const  
*Get min, max and average values.*
- void [findMax](#) ([Point2D](#) &p, T &maxval) const  
*find point of max activity and also what that max activity is*
- void [findMin](#) ([Point2D](#) &p, T &minval) const  
*find point of min activity and also what that min activity is*
- double [getMean](#) () const  
*compute mean pixel value over image (promoted to double)*
- double [getStdev](#) () const  
*compute standard deviation of pixel values over image*
- T & [operator\[\]](#) (const int index)  
*Access image elements through C array index interface.*
- const T & [operator\[\]](#) (const int index) const  
*Access image elements through C array index interface.*
- const T & [getVal](#) (const int index) const  
*Get pixel value at index in image.*
- const T & [getVal](#) (const int x, const int y) const  
*Get pixel value at (x, y) in image.*
- const T & [getVal](#) (const [Point2D](#) &p) const  
*Get pixel value at specified coordinates in image.*
- template<class T2> void [getVal](#) (const int x, const int y, T2 &val) const  
*Get value at (x,y), put in val rather than return it.*
- const T [getValInterp](#) (const float x, const float y) const  
*Get pixel value at (x, y) in image with bilinear interpolation.*
- T [getSample](#) (const [Point2D](#) &center, const int radius)  
*Get a sample which is the max value within a circular aperture.*
- template<class T2> void [setVal](#) (const int index, const T2 &value)  
*Set value in image at index.*
- template<class T2> void [setVal](#) (const [Point2D](#) &p, const T2 &value)  
*Set value in image at [Point2D](#).*

- `template<class T2> void setVal (const int x, const int y, const T2 &value)`  
*Set value in image at (x, y).*
- `void setValMask (const Image< byte > &mask, const T val)`  
*Set values at all locations where mask is non-zero.*
- `void replaceVal (const T &from, const T &to)`  
*Replace all occurrences of a value 'from' by another value 'to'.*
- `const T * getArrayPtr () const`  
*Returns read-only (const) pointer to internal image array.*
- `T * getArrayPtr ()`  
*Returns read/write (non-const) pointer to internal image array.*
- `bool coordsOk (const Point2D &P) const`  
*Test whether point falls inside array boundaries.*
- `bool coordsOk (const int i, const int j) const`  
*Test whether point falls inside array boundaries.*
- `bool rectangleOk (const Rectangle &rect) const`  
*Test whether rectangle fits in image.*

### Operator overloads and basic image manipulations

Note that many other image manipulation functions are available in [Image\\_ColorOps.H](#), [Image\\_FilterOps.H](#), [Image\\_IO.H](#), [Image\\_MathOps.H](#), [Image\\_Omni.H](#), [Image\\_ShapeOps.H](#), and [Image\\_Transforms.H](#).

- `Image< T > & operator+= (const T &val)`  
*Add constant to image, clamp result as necessary.*
- `Image< T > & operator-= (const T &val)`  
*Subtract constant from image, clamp result as necessary.*
- `Image< T > & operator *= (const T &val)`  
*Multiply image by constant, clamp result as necessary.*
- `Image< T > & operator/= (const T &val)`  
*Divide image by constant, clamp result as necessary.*
- `template<class T2> Image< T > & operator+= (const Image< T2 > &A)`  
*Add image to image, clamp result as necessary.*
- `template<class T2> Image< T > & operator-= (const Image< T2 > &A)`

*Subtract image from image, clamp result as necessary.*

- `template<class T2> Image< T > & operator *= (const Image< T2 > &A)`  
*Multiply image by image, point-by-point, clamp result as necessary.*
- `template<class T2> Image< T > & operator /= (const Image< T2 > &A)`  
*Divide image by image, point-by-point, clamp result as necessary.*
- `template<class T2> Image< typename promote_trait< T, T2 >::TP > operator+ (const T2 &val) const`  
*Add scalar to each point in \*this and return result.*
- `template<class T2> Image< typename promote_trait< T, T2 >::TP > operator- (const T2 &val) const`  
*Subtract scalar from each point in \*this and return result.*
- `template<class T2> Image< typename promote_trait< T, T2 >::TP > operator * (const T2 &val) const`  
*Multiply scalar each point in \*this by scalar and return result.*
- `template<class T2> Image< typename promote_trait< T, T2 >::TP > operator/ (const T2 &val) const`  
*Divide each point in \*this by scalar and return result.*
- `template<class T2> Image< typename promote_trait< T, T2 >::TP > operator+ (const Image< T2 > &img) const`  
*Point-wise add img to image and return result.*
- `template<class T2> Image< typename promote_trait< T, T2 >::TP > operator- (const Image< T2 > &img) const`  
*Point-wise subtract img from \*this and return result.*
- `template<class T2> Image< typename promote_trait< T, T2 >::TP > operator * (const Image< T2 > &img) const`  
*Point-wise multiply \*this by img and return result.*
- `template<class T2> Image< typename promote_trait< T, T2 >::TP > operator/ (const Image< T2 > &img) const`  
*Point-wise divide \*this by img and return result.*
- `void addScaled (const Image< T > &b)`  
*Scale b to my size and add it to me (using bilinear interpolation).*
- `void addDownsized (const Image< T > &b)`  
*Use pyramid to downsize b and add to me.*
- `void addWeighted (const Image< T > &b, const float coeff)`

*this += b \* coeff*

- void [square](#) ()  
*Take square of all elements.*
- void [normalize](#) (const T nmin, const T nmax)  
*Normalize values between nmin and nmax.*
- void [activityNormalize](#) (const T nmax=255.0)  
*Normalize image by dividing by total activity.*
- void [varianceNormalize](#) (const T mi, const T ma)  
*Normalize between mi and ma and multiply by variance.*
- void [luminanceNormalize](#) (const [Image](#)< T > &lum, const T thresh)  
*Normalize my values by those in the lum image.*
- void [clamp](#) (const T cmin, const T cmax)  
*Saturate values < cmin to cmin and > cmax to cmax.*
- void [rectify](#) ()  
*Saturate values < 0.*
- void [splitPosNeg](#) ([Image](#)< T > &pos, [Image](#)< T > &neg)  
*Put all values >= 0 into pos and the negated of all vals <= 0 into neg.*
- void [lowThresh](#) (const T thresh, const T val=T())  
*Cut values < thresh and replace them by val.*
- void [lowThreshAbs](#) (const T thresh, const T val=T())  
*Cut values whose abs is < thresh and replace them by val.*
- void [attenuateBorders](#) (int size)  
*Progressive attenuation of borders by "size" pixels.*
- void [setBorders](#) (const int size, const T value=T())  
*set borders of "size" pixels to given value*
- void [paste](#) (const [Image](#)< T > &img, const [Point2D](#) &pos)  
*Paste image into \*this at given position.*
- void [pasteGabor](#) (const [Image](#)< T > &gabor, const [Point2D](#) &pos, const T background)  
*Paste a gabor into an image area. Account for overlap.*
- void [embed](#) (const [Image](#)< T > &img, const [Rectangle](#) &r, const T background, const bool keep\_aspect=true)

*Embed image into \*this with position & size given in rectangle.*

- void [wilsonNormalize](#) (const float H, const float eps)  
*Wilson non-linearity.*
- void [sigmoid](#) (const float g, const float h, const float s)  
*Pass image through sigmoid:  $f(x) = x^g / (s + x^h)$ .*
- void [sigmoid](#) (const float s)  
*Pass image through sigmoid:  $f(x) = x^2 / (s + x^2)$ .*
- void [clear](#) (const T &val=T())  
*clear contents (or set to given value)*
- void [getVector](#) (std::vector< std::vector< T > > \*V) const  
*return a double nested vector that represents data (see also =operator)*
- void [getVector](#) (std::vector< std::vector< T \* > > \*V) const  
*return a double nested vector that represents data (see also =operator)*
- void [getVectorRow](#) (std::vector< T > \*V, int row) const  
*return a single row of an image as a vector*
- void [getVectorRow](#) (std::vector< T \* > \*V, int row) const  
*return a single row of an image as a vector*
- void [getVectorColumn](#) (std::vector< T > \*V, int column) const  
*return a single column of an image as a vector*
- void [getVectorColumn](#) (std::vector< T \* > \*V, int column) const  
*return a single column of an image as a vector*
- void [retainMax](#) (const int num)  
*retain the num largest values and set the other entries to zero*

### Drawing functions

- int [emptyArea](#) () const  
*Tells how many pixels are zero.*
- void [drawDisk](#) (const [Point2D](#) &center, const int radius, const T value)  
*Draws a disk with color "value" in image.*
- int [drawDiskCheckTarget](#) ([Image](#)< T > &mask, const [Point2D](#) &center, const int radius, const T value, const T targetval, const T floodval)  
*Draws a disk in image and check if it hits a target.*



- void **drawLine** (const **Point2D** &p1, const **Point2D** &p2, const T col, const int rad=1)  
*Draw a line from p1 to p2 of color col and pixel radius rad.*
- void **drawRect** (const **Rectangle** &rect, const T col, const int rad=1)  
*Draw a rectangle of color col and pixel radius rad.*
- void **drawCross** (const **Point2D** &p, const T col, const int siz=3, const int rad=1)  
*Draw a cross of color col, half-size siz and pixel radius rad.*
- void **drawPatch** (const **Point2D** &p, const int siz, const T col)  
*Draw a square patch of color col and size siz.*
- void **drawCircle** (const **Point2D** &p, const int radius, const T col, const int rad=1)  
*Draw a circle of color col and radius radius, with pixel radius rad.*
- void **drawArrow** (const **Point2D** &p1, const **Point2D** &p2, const T col, const int rad=1)  
*Draw an arrow from p1 to p2 with pixel radius rad.*
- void **drawGrid** (const int spacingX, const int spacingY, const int thickX, const int thickY, const T col)  
*Draw a grid.*
- void **writeText** (const **Point2D** &top\_left, const char \*text, const T col=T(), const T bgcol=T(255))  
*Write some text; only ASCII chars between 32 and 126 are supported.*
- void **drawPoint** (int X, int Y, T pix)  
*draw a dot which is a point with 4 connected points*

### Miscellaneous functions

- int **flood** (**Image**< T > &dest, const **Point2D** &seed, const T thresh, const T val) const  
*Floods object >= thresh with val from seed and returns area.*
- int **countParticles** (const T thresh) const  
*How many particles >= thresh ?*
- int **countThresh** (const T thresh, const bool absol=true) const  
*Counts how many pixels are > thresh (in absolute value if absol true).*

- void [speckleNoise](#) (const float thresh, const int size, const T noise\_color, bool random\_color=false)  
*Add speckle noise to array; thresh = 1 for 100% noise.*
- void [addBGnoise](#) (const float range)  
*Add background noise to image.*
- void [addBGnoise2](#) (const float range)  
*Add background noise to image.*

### Functions for testing/debugging only

- bool [hasSameData](#) (const [Image](#)< T > &b) const  
*For testing/debugging only.*
- long [refCount](#) () const throw ()  
*For testing/debugging only.*

## 10.122.1 Detailed Description

**template<class T> class [Image](#)< T >**

Generic image template class.

This is a generic image template class. In principle, it can handle grayscale as well as color or multispectral images. Not all methods will work with both types, though (e.g., some methods need [luminance\(\)](#) to be defined in the base type [will not work on canonical scalar types], and some other methods perform arithmetic comparisons on the base type [will not work with color pixels]).

Note also that many other image manipulation functions are defined as non-member functions in [Image\\_ColorOps.H](#), [Image\\_FilterOps.H](#), [Image\\_IO.H](#), [Image\\_MathOps.H](#), [Image\\_Omni.H](#), [Image\\_ShapeOps.H](#), and [Image\\_Transforms.H](#).

Definition at line 71 of file [Image.H](#).

## 10.122.2 Member Typedef Documentation

**10.122.2.1 template<class T> typedef const T\* [Image](#)< T >::const\_iterator**

Read-only iterator.

Definition at line 227 of file [Image.H](#).

Referenced by Image< T >::addScaled(), Image< T >::addWeighted(), Image< T >::begin(), Image< T >::countThresh(), Image< T >::embed(), Image< T >::end(), Image< T >::findMax(), Image< T >::findMin(), Image< T >::getMinMax(), Image< T >::getMinMaxAvg(), Image< T >::getStddev(), Image< T >::getValInterp(), Image< T >::getVector(), Image< T >::getVectorRow(), Image< T >::Image(), Image< T >::isFinite(), Image< T >::luminanceNormalize(), Image< T >::operator\*(), Image< T >::operator\*=( ), Image< T >::operator+(), Image< T >::operator+=( ), Image< T >::operator-(), Image< T >::operator-=( ), Image< T >::operator/(), Image< T >::operator/=( ), Image< T >::operator=(), Image< T >::paste(), Image< T >::pasteGabor(), Image< T >::retainMax(), Image< T >::setValMask(), and Image< T >::splitPosNeg().

### 10.122.2.2 template<class T> typedef T\* Image< T >::iterator

Read/write iterator.

Definition at line 225 of file Image.H.

Referenced by Image< T >::activityNormalize(), Image< T >::addBGnoise(), Image< T >::addBGnoise2(), Image< T >::addScaled(), Image< T >::addWeighted(), Image< T >::attenuateBorders(), Image< T >::beginw(), Image< T >::clamp(), Image< T >::clear(), Image< T >::embed(), Image< T >::endw(), Image< T >::flood(), Image< T >::getVector(), Image< T >::getVectorColumn(), Image< T >::getVectorRow(), Image< T >::Image(), Image< T >::lowThresh(), Image< T >::lowThreshAbs(), Image< T >::luminanceNormalize(), Image< T >::normalize(), Image< T >::operator\*(), Image< T >::operator\*=( ), Image< T >::operator+(), Image< T >::operator+=( ), Image< T >::operator-(), Image< T >::operator-=( ), Image< T >::operator/(), Image< T >::operator/=( ), Image< T >::operator=(), Image< T >::paste(), Image< T >::pasteGabor(), Image< T >::rectify(), Image< T >::replaceVal(), Image< T >::retainMax(), Image< T >::setBorders(), Image< T >::setValMask(), Image< T >::sigmoid(), Image< T >::splitPosNeg(), Image< T >::square(), and Image< T >::wilsonNormalize().

## 10.122.3 Constructor & Destructor Documentation

### 10.122.3.1 template<class T> Image< T >::Image (const T \* *inarray*, int *width*, int *height*) [inline]

Construct from C array.

Build from C array; an internal copy of the C array will be allocated, so the C array can (and should) be freed without affecting the Image.

Definition at line 671 of file Image.H.

**10.122.3.2** `template<class T> Image< T >::Image (const T * inarray, const Dims & dims) [inline]`

Construct from C array.

Build from C array; an internal copy of the C array will be allocated, so the C array can (and should) be freed without affecting the Image.

Definition at line 677 of file Image.H.

**10.122.3.3** `template<class T> Image< T >::Image (int width, int height, InitPolicy init) [inline]`

Allocates memory for given size, and optionally zero-clear that memory.

Definition at line 683 of file Image.H.

**10.122.3.4** `template<class T> Image< T >::Image (const Dims & dims, InitPolicy init) [inline, explicit]`

Constructor that only allocates memory for given size.

Definition at line 689 of file Image.H.

**10.122.3.5** `template<class T> Image< T >::Image () [inline]`

Construct an empty (0-by-0) image (useful for arrays of Images).

Definition at line 695 of file Image.H.

**10.122.3.6** `template<class T> Image< T >::Image (const Image< T > & A) [inline]`

Copy constructor.

e.g.:

```
Image<byte> im(other);  
// or  
Image<byte> im = other; // with other also of type Image<byte>
```

Definition at line 701 of file Image.H.

### 10.122.3.7 `template<class T> template<class T2> Image< T >::Image (const Image< T2 > & A) [inline]`

Conversion copy constructor.

e.g.:

```
Image<byte> im(other);
// or
Image<byte> im = other; // with other of type Image<float>
```

Definition at line 707 of file Image.H.

References Image< T >::begin(), Image< T >::beginw(), Image< T >::const\_iterator, Image< T >::endw(), Image< T >::iterator, and NO\_INIT.

### 10.122.3.8 `template<class T> Image< T >::~~Image () [inline]`

Destructor.

Definition at line 827 of file Image.H.

## 10.122.4 Member Function Documentation

### 10.122.4.1 `template<class T> void Image< T >::activityNormalize (const T nmax = 255.0)`

Normalize image by dividing by total activity.

Definition at line 199 of file Image\_MonoOnly.C.

References Image< T >::beginw(), Image< T >::endw(), and Image< T >::iterator.

### 10.122.4.2 `template<class T> void Image< T >::addBGnoise (const float range)`

Add background noise to image.

[Range](#) should be max value for any image of the type to which we add noise

Definition at line 548 of file Image\_MonoOnly.C.

References ASSERT, Image< T >::beginw(), Image< T >::getHeight(), Image< T >::getWidth(), Image< T >::initialized(), and Image< T >::iterator.

Referenced by computeCMAP(), and SingleChannel::postProcessMap().

#### 10.122.4.3 **template<class T> void [Image](#)< T >::addBGnoise2 (const float *range*)**

Add background noise to image.

[Range](#) should be max value for any image of the type to which we add noise similar to addBGnoise, but eliminates BNG.. constant

Definition at line 569 of file Image\_MonoOnly.C.

References ASSERT, Image< T >::beginw(), Image< T >::getHeight(), Image< T >::getWidth(), Image< T >::initialized(), and Image< T >::iterator.

#### 10.122.4.4 **template<class T> void [Image](#)< T >::addDownsized (const [Image](#)< T > & *b*)**

Use pyramid to downsize b and add to me.

Definition at line 173 of file Image\_MonoOnly.C.

References downSize(), and Image< T >::getDims().

#### 10.122.4.5 **template<class T> void [Image](#)< T >::addScaled (const [Image](#)< T > & *b*)**

Scale b to my size and add it to me (using bilinear interpolation).

Definition at line 111 of file Image.C.

References ASSERT, Image< T >::begin(), Image< T >::beginw(), Image< T >::const\_iterator, Image< T >::getHeight(), Image< T >::getWidth(), Image< T >::initialized(), Image< T >::isSameSize(), and Image< T >::iterator.

#### 10.122.4.6 **template<class T> void [Image](#)< T >::addWeighted (const [Image](#)< T > & *b*, const float *coeff*)**

this += b \* coeff

Definition at line 162 of file Image.C.

References ASSERT, Image< T >::begin(), Image< T >::beginw(), Image< T >::const\_iterator, Image< T >::end(), Image< T >::getDims(), Image< T >::initialized(), Image< T >::iterator, and NO\_INIT.

Referenced by SoxChannel::getNonlinearResponse().

#### 10.122.4.7 `template<class T> void Image< T >::attach (T * array, const int width, const int height) [inline]`

Use existing memory.

This is potentially dangerous and should really be avoided. The only case where this really is useful is to attach an image to an existing memory segment that is shared or into which data is streaming via DMA.

Definition at line 811 of file Image.H.

References WRITE\_THRU.

#### 10.122.4.8 `template<class T> void Image< T >::attenuateBorders (int size)`

Progressive attenuation of borders by "size" pixels.

Definition at line 199 of file Image.C.

References ASSERT, Image< T >::beginw(), Image< T >::getDims(), Dims::h(), Image< T >::initialized(), Image< T >::iterator, and Dims::w().

Referenced by maxNormalizeFancyFast().

#### 10.122.4.9 `template<class T> Image< T >::const_iterator Image< T >::begin () const [inline]`

Returns a read-only iterator to the beginning of the image data.

Definition at line 884 of file Image.H.

References Image< T >::const\_iterator.

Referenced by abs(), absDiff(), Image< T >::addScaled(), Image< T >::addWeighted(), average(), averageWeighted(), avgOrient(), binaryReverse(), chamfer34(), clampedDiff(), colorStain(), SurpriseModelSP::combineFrom(), SurpriseModelSG::combineFrom(), contour2D(), contour2D8(), convolveHmax(), corrcoeff(), corrpach(), Image< T >::countThresh(), dilateImg(), distance(), distDegrade(), VarianceChannel::doInput(), EntropyChannel::doInput(), Image< T >::embed(), erodeImg(), exp(), featurePoolHmax(), Image< T >::findMax(), Image< T >::findMin(), SurpriseImage< T >::getMean(), Image< T >::getMinMax(), Image< T >::getMinMaxAvg(), getRGBY(), Image< T >::getStdev(), Jet< T >::getVal(), Image< T >::getVal(), Image< T >::getValInterp(), SurpriseImage< T >::getVar(), Image< T >::getVector(), Image< T >::getVectorColumn(), Image< T >::getVectorRow(), getYIQ(), getYUV(), hmaxActivation(), Image< T >::Image(), Image2mexArray(), SurpriseImage< T >::init(), SaliencyMapStd::input(), inverse(), Image< T >::isFinite(), junctionFilterFull(), junctionFilterPartial(), learningCoeff(), main(), makeBinary(), makeRGB(), negexp(), normalizeRGPolar(), Image< T >::operator\*(), Image< T >::operator\*=( ), Image< T >::operator+( ), Image< T >::operator+=( ),

Image< T >::operator-(), Image< T >::operator=(), Image< T >::operator/(), Image< T >::operator+=(), Image< T >::operator\*=(), Image< T >::operator[ ](), optConvolve(), overlay(), overlayStain(), Image< T >::paste(), Image< T >::pasteGabor(), pasteImage(), rangeOf(), remapRange(), replaceVals(), Image< T >::retainMax(), RMSerr(), BitObject::setMaxMinAvgIntensity(), Image< T >::setValMask(), setYUV(), spatialPoolMax(), Image< T >::splitPosNeg(), sqrt(), squared(), stain(), sum(), sumXY(), SurpriseImage< T >::surprise(), takeMax(), takeMin(), toPower(), toRGB(), transpose(), FOEstimator::updateFOE(), writeBinImage(), and writeImageToStream().

#### 10.122.4.10 `template<class T> Image< T >::iterator Image< T >::beginw ()` [inline]

Returns a read-write iterator to the beginning of the image data.

Definition at line 892 of file Image.H.

References Image< T >::iterator.

Referenced by abs(), absDiff(), Image< T >::activityNormalize(), Image< T >::addBGnoise(), Image< T >::addBGnoise2(), Image< T >::addScaled(), Image< T >::addWeighted(), Image< T >::attenuateBorders(), average(), averageWeighted(), avgOrient(), binaryReverse(), chamfer34(), Image< T >::clamp(), clampedDiff(), Image< T >::clear(), colorStain(), contour2D(), contour2D8(), convolveHmax(), copy(), dilateImg(), distDegrade(), dogFilter(), BitObject::drawShape(), Image< T >::embed(), erodeImg(), exp(), eye(), fixationMask(), fromARGB(), fromBGR(), fromBGRA(), fromMono(), fromRGB555(), fromRGB555bs(), fromRGB565(), fromRGB565bs(), fromYUV410P(), fromYUV411(), fromYUV411bs(), fromYUV411P(), fromYUV420P(), fromYUV422(), fromYUV422bs(), fromYUV422P(), fromYUV444(), fromYUV444bs(), fromYUV444P(), gabor2(), gaborFilter(), gaborFilter2(), gaborFilter3(), gaussian2D(), getComponents(), SurpriseImage< T >::getMean(), getRGBY(), WinnerTakeAllStd::getV(), SaliencyMapStd::getV(), SurpriseImage< T >::getVar(), getYIQ(), getYUV(), grating(), highThresh(), hmaxActivation(), hueDistance(), Image< T >::Image(), SurpriseImage< T >::init(), inverse(), junctionFilterFull(), junctionFilterPartial(), longRangeExcFilter(), Image< T >::lowThresh(), Image< T >::lowThreshAbs(), luminance(), Image< T >::luminanceNormalize(), luminanceNormalize(), main(), makeBinary(), makeRGB(), matrixMult(), mexArray2Image(), mexArray2RGBImage(), negexp(), SurpriseImage< T >::neighborhoods(), Image< T >::normalize(), normalizeRGPolar(), Image< T >::operator \*(), Image< T >::operator +=(), Image< T >::operator+(), Image< T >::operator-=(), Image< T >::operator/(), Image< T >::operator\*=(), Image< T >::operator=(), Image< T >::operator[ ](), optConvolve(), overlay(), overlayStain(), Image< T >::paste(), Image< T >::pasteGabor(), pasteImage(), readBinImage(), readImageFromStream(), Image< T >::rectify(), remapRange(), Image< T >::replaceVal(), replaceVals(), SurpriseImage< T >::reset(), SurpriseImage< T



>::resetUpdFac(), Image< T >::setBorders(), Jet< T >::setVal(), Image< T >::setVal(), Image< T >::setValMask(), setYUV(), Image< T >::sigmoid(), spatialPoolMax(), Image< T >::splitPosNeg(), sqrt(), Image< T >::square(), squared(), squash(), stain(), SurpriseImage< T >::surprise(), takeMax(), takeMin(), toPower(), toRGB(), transpose(), and Image< T >::wilsonNormalize().

#### 10.122.4.11 `template<class T> void Image< T >::clamp (const T cmin, const T cmax) [inline]`

Saturate values < cmin to cmin and > cmax to cmax.

Definition at line 1306 of file Image.H.

References Image< T >::beginw(), Image< T >::endw(), and Image< T >::iterator.

Referenced by contrastModulate(), SimulationViewerStd::drawMegaCombo(), TaskRelevanceMapKillStatic::evolve(), ShapeEstimator::getCumSmoothMask(), ShapeEstimator::getNegCumSmoothMask(), TaskRelevanceMapKillN::getV(), TaskRelevanceMapKillStatic::getV(), TaskRelevanceMapStd::getV(), TaskRelevanceMapKillN::input(), main(), and maxNormalizeFancy().

#### 10.122.4.12 `template<class T> void Image< T >::clear (const T & val = T()) [inline]`

clear contents (or set to given value)

Definition at line 1319 of file Image.H.

References Image< T >::beginw(), Image< T >::endw(), Image< T >::getDims(), Image< T >::iterator, and NO\_INIT.

Referenced by SimulationViewerEyeMvt2::attentionShift(), PsychoDisplay::displayText(), ScorrChannel::doInput(), VisualEventSet::drawTokens(), SoxChannel::getNonlinearResponse(), VisualCortexEyeMvt::getOutput(), SingleChannelSurprise::getSurpriseMap(), SimulationViewerEyeMvt2::getTraj(), SimulationViewerCompress::getTraj(), VisualBuffer::input(), Brain::obtainOutput(), ones(), Image< T >::resize(), and SurpriseMap< T >::surprise().

#### 10.122.4.13 `template<class T> bool Image< T >::coordsOk (const int i, const int j) const [inline]`

Test whether point falls inside array boundaries.

Definition at line 1089 of file Image.H.

References Image< T >::getHeight(), and Image< T >::getWidth().

#### 10.122.4.14 `template<class T> bool Image< T >::coordsOk (const Point2D & P) const [inline]`

Test whether point falls inside array boundaries.

Definition at line 1082 of file Image.H.

References Image< T >::getHeight(), Image< T >::getWidth(), Point2D::i, and Point2D::j.

Referenced by SurpriseModelSP::combineFrom(), SurpriseModelSG::combineFrom(), Image< T >::drawCircle(), Image< T >::drawDisk(), Image< T >::drawDiskCheckTarget(), BitObject::drawOutline(), SimulationViewerEyeMvt2::eyeMovement(), Image< T >::flood(), floodClean(), floodCleanBB(), PyrFoveator::foveate(), BlurFoveator::foveate(), ImageSpring< T >::getNeighbor(), Image< T >::getSample(), Image< T >::getVal(), Image< T >::getValInterp(), main(), and Image< T >::setVal().

#### 10.122.4.15 `template<class T> int Image< T >::countParticles (const T thresh) const`

How many particles  $\geq$  thresh ?

Definition at line 499 of file Image\_MonoOnly.C.

References Image< T >::flood(), Image< T >::getHeight(), Image< T >::getVal(), Image< T >::getWidth(), Point2D::i, and Point2D::j.

Referenced by main().

#### 10.122.4.16 `template<class T> int Image< T >::countThresh (const T thresh, const bool absol = true) const`

Counts how many pixels are  $>$  thresh (in absolute value if absol true).

Definition at line 522 of file Image\_MonoOnly.C.

References ASSERT, Image< T >::begin(), Image< T >::const\_iterator, Image< T >::end(), and Image< T >::initialized().

Referenced by infoMeasure(), and main().

#### 10.122.4.17 `template<class T> void Image< T >::detach () [inline]`

Detach previously `attach()`'ed image.

The main purpose of `detach()` is to make sure that this Image object does not continue to point at `attach()`'ed memory after that memory has been freed. Nevertheless, this function is not strictly necessary to ensure correct memory handling, since the Image

destructor will only try to free memory if that memory is owned (i.e., not `attach()`'ed). All that it does is release any association with a previously `attach()`'ed memory block, by setting the current image to a new empty (zero-by-zero) `ArrayData` object. We assume that attached memory will be destroyed later.

Definition at line 819 of file Image.H.

#### 10.122.4.18 `template<class T> void Image< T >::drawArrow (const Point2D & p1, const Point2D & p2, const T col, const int rad = 1)`

Draw an arrow from p1 to p2 with pixel radius rad.

Definition at line 545 of file Image.C.

References `Image< T >::drawLine()`, `Image< T >::getHeight()`, `Image< T >::getWidth()`, `Point2D::i`, and `Point2D::j`.

Referenced by `SimulationViewerStd::drawMotionSixPack()`, and `main()`.

#### 10.122.4.19 `template<class T> void Image< T >::drawCircle (const Point2D & p, const int radius, const T col, const int rad = 1)`

Draw a circle of color col and radius radius, with pixel radius rad.

Definition at line 519 of file Image.C.

References `ASSERT`, `Image< T >::coordsOk()`, `Image< T >::drawDisk()`, `Point2D::i`, `Image< T >::initialized()`, `Point2D::j`, `Image< T >::setVal()`, `SQR()`, and `sqrt()`.

Referenced by `SimulationViewerEyeMvt2::getTraj()`, `SimulationViewerEyeMvt::getTraj()`, and `main()`.

#### 10.122.4.20 `template<class T> void Image< T >::drawCross (const Point2D & p, const T col, const int siz = 3, const int rad = 1)`

Draw a cross of color col, half-size siz and pixel radius rad.

Definition at line 482 of file Image.C.

References `ASSERT`, `CLAMP()`, `Image< T >::drawLine()`, `Point2D::i`, `Image< T >::initialized()`, and `Point2D::j`.

Referenced by `ImageSpring< T >::getClusteredImage()`.

#### 10.122.4.21 `template<class T> void Image< T >::drawDisk (const Point2D & center, const int radius, const T value)`

Draws a disk with color "value" in image.

Definition at line 417 of file Image.C.

References ASSERT, Image< T >::coordsOk(), Point2D::i, Image< T >::initialized(), Point2D::j, Image< T >::setVal(), SQR(), and sqrt().

Referenced by SimulationViewerEyeMvt2::attentionShift(), RetinaStd::doGetOutput(), Image< T >::drawCircle(), Image< T >::drawLine(), SimulationViewerEyeMvt2::getTraj(), SimulationViewerCompress::getTraj(), VisualBuffer::inhibit(), VisualBuffer::input(), and main().

**10.122.4.22** `template<class T> int Image< T >::drawDiskCheckTarget (Image< T > & mask, const Point2D & center, const int radius, const T value, const T targetval, const T floodval)`

Draws a disk in image and check if it hits a target.

**Parameters:**

*mask* input binary target mask to draw into  
*center* coordinates of disk center  
*radius* radius of disk, in pixels  
*value* pixel value used to draw the disk  
*targetval* value with which the targets are represented  
*floodval* value to flood target with if target hit

**Returns:**

number of targets hit and flooded

Definition at line 397 of file Image\_MonoOnly.C.

References ASSERT, Image< T >::coordsOk(), Image< T >::flood(), Image< T >::getVal(), Point2D::i, Image< T >::initialized(), Point2D::j, Image< T >::setVal(), SQR(), and sqrt().

**10.122.4.23** `template<class T> void Image< T >::drawGrid (const int spacingX, const int spacingY, const int thickX, const int thickY, const T col)`

Draw a grid.

Definition at line 566 of file Image.C.

References ASSERT, Image< T >::getHeight(), Image< T >::getWidth(), Image< T >::initialized(), and Image< T >::setVal().

Referenced by SimulationViewerStd::drawMotionSixPack(), SimulationViewerStd::getTraj(), SimulationViewerEyeMvt::getTraj(), SimulationViewerCompress::getTraj(), main(), and makeImageArray().

**10.122.4.24    template<class T> void Image< T >::drawLine (const Point2D & p1, const Point2D & p2, const T col, const int rad = 1)**

Draw a line from p1 to p2 of color col and pixel radius rad.

Definition at line 435 of file Image.C.

References ABS(), ASSERT, Image< T >::drawDisk(), Point2D::i, Image< T >::initialized(), Point2D::j, and SGN().

Referenced by Image< T >::drawArrow(), Image< T >::drawCross(), Image< T >::drawRect(), SimulationViewerEyeMvt2::getTraj(), SimulationViewerEyeMvt::getTraj(), SimulationViewerCompress::getTraj(), main(), and warp3D().

**10.122.4.25    template<class T> void Image< T >::drawPatch (const Point2D & p, const int siz, const T col)**

Draw a square patch of color col and size siz.

Definition at line 505 of file Image.C.

References ASSERT, CLAMP(), Point2D::i, Image< T >::initialized(), Point2D::j, and Image< T >::setVal().

Referenced by SimulationViewerCompress::getTraj().

**10.122.4.26    template<class T> void Image< T >::drawPoint (int X, int Y, T pix)**

draw a dot which is a point with 4 connected points

this is between the setVal method and drawCircle method for illustrating points

Definition at line 608 of file Image.C.

References Image< T >::getHeight(), Image< T >::getWidth(), Image< T >::setVal(), and Y.

**10.122.4.27    template<class T> void Image< T >::drawRect (const Rectangle & rect, const T col, const int rad = 1)**

Draw a rectangle of color col and pixel radius rad.

Definition at line 470 of file Image.C.

References ASSERT, Rectangle::bottom(), Image< T >::drawLine(), Image< T >::initialized(), Rectangle::left(), Image< T >::rectangleOk(), Rectangle::right(), and Rectangle::top().

Referenced by `SimulationViewerStd::drawMegaCombo()`, `SimulationViewerStd::getTraj()`, and `main()`.

**10.122.4.28** `template<class T> void Image< T >::embed (const Image< T > & img, const Rectangle & r, const T background, const bool keep_aspect = true)`

Embed image into \*this with position & size given in rectangle.

Definition at line 340 of file `Image.C`.

References `ASSERT`, `Image< T >::begin()`, `Image< T >::beginw()`, `Image< T >::const_iterator`, `Image< T >::getHeight()`, `Image< T >::getWidth()`, `Rectangle::height()`, `Image< T >::initialized()`, `Image< T >::iterator`, `Rectangle::left()`, `Image< T >::rectangleOk()`, `Rectangle::top()`, and `Rectangle::width()`.

Referenced by `SimulationViewerStd::drawMegaCombo()`, `main()`, and `FrameSeries::resizeImage()`.

**10.122.4.29** `template<class T> int Image< T >::emptyArea () const`

Tells how many pixels are zero.

Definition at line 390 of file `Image_MonoOnly.C`.

References `Image< T >::end()`.

**10.122.4.30** `template<class T> Image< T >::const_iterator Image< T >::end () const [inline]`

Returns a read-only iterator to one-past-the-end of the image data.

Definition at line 888 of file `Image.H`.

References `Image< T >::const_iterator`.

Referenced by `abs()`, `Image< T >::addWeighted()`, `colorStain()`, `SurpriseModelSP::combineFrom()`, `SurpriseModelSG::combineFrom()`, `corrcoef()`, `Image< T >::countThresh()`, `distance()`, `Image< T >::emptyArea()`, `exp()`, `SurpriseImage< T >::getMean()`, `Image< T >::getMean()`, `Image< T >::getMinMax()`, `Image< T >::getMinMaxAvg()`, `Image< T >::getStddev()`, `SurpriseImage< T >::getVar()`, `SaliencyMapStd::input()`, `inverse()`, `Image< T >::isFinite()`, `learningCoeff()`, `neg-exp()`, `Image< T >::operator*()`, `Image< T >::operator+()`, `Image< T >::operator-()`, `Image< T >::operator/()`, `rangeOf()`, `RMSerr()`, `Image< T >::splitPosNeg()`, `sqrt()`, `squared()`, `sum()`, and `toPower()`.

#### 10.122.4.31 `template<class T> Image< T >::iterator Image< T >::endw ()` [inline]

Returns a read-write iterator to one-past-the-end of the image data.

Definition at line 896 of file Image.H.

References Image< T >::iterator.

Referenced by absDiff(), Image< T >::activityNormalize(), average(), average-Weighted(), binaryReverse(), Image< T >::clamp(), clampedDiff(), Image< T >::clear(), fromARGB(), fromBGR(), fromBGRa(), fromMono(), fromRGB555(), fromRGB555bs(), fromRGB565(), fromRGB565bs(), fromYUV411(), fromYUV411bs(), fromYUV422(), fromYUV422bs(), fromYUV444(), fromYUV444bs(), WinnerTakeAllStd::getV(), highThresh(), Image< T >::Image(), SurpriseImage< T >::init(), Image< T >::lowThresh(), Image< T >::lowThreshAbs(), luminance(), Image< T >::luminanceNormalize(), makeBinary(), makeRGB(), Image< T >::normalize(), normalizeRGPolar(), Image< T >::operator \*(), Image< T >::operator +(), Image< T >::operator -(), Image< T >::operator /(), Image< T >::operator =(), overlay(), overlayStain(), readImageFromStream(), Image< T >::rectify(), remapRange(), Image< T >::replaceVal(), replaceVals(), SurpriseImage< T >::reset(), SurpriseImage< T >::resetUpdFac(), Image< T >::setValMask(), setYUV(), Image< T >::sigmoid(), Image< T >::square(), stain(), SurpriseImage< T >::surprise(), takeMax(), takeMin(), toRGB(), and Image< T >::wilsonNormalize().

#### 10.122.4.32 `template<class T> void Image< T >::findMax (Point2D & p, T & maxval) const`

find point of max activity and also what that max activity is

Definition at line 113 of file Image\_MonoOnly.C.

References ASSERT, Image< T >::begin(), Image< T >::const\_iterator, Image< T >::getHeight(), Image< T >::getWidth(), Point2D::i, Image< T >::initialized(), and Point2D::j.

Referenced by TemplateMatchChannel::findBestMatch(), findPeaks(), goodness\_map(), and maxNormalizeLandmark().

#### 10.122.4.33 `template<class T> void Image< T >::findMin (Point2D & p, T & minval) const`

find point of min activity and also what that min activity is

Definition at line 129 of file Image\_MonoOnly.C.

References ASSERT, Image< T >::begin(), Image< T >::const\_iterator, Image< T

`>::getHeight()`, `Image< T >::getWidth()`, `Point2D::i`, `Image< T >::initialized()`, and `Point2D::j`.

Referenced by `VisualBuffer::findMostInterestingTarget()`.

#### 10.122.4.34 `template<class T> int Image< T >::flood (Image< T > & dest, const Point2D & seed, const T thresh, const T val) const`

Floods object  $\geq$  thresh with val from seed and returns area.

Definition at line 430 of file `Image_MonoOnly.C`.

References `ASSERT`, `Image< T >::coordsOk()`, `Image< T >::getDims()`, `Image< T >::getSize()`, `Image< T >::getVal()`, `Image< T >::getWidth()`, `Point2D::i`, `Image< T >::initialized()`, `Image< T >::iterator`, `Point2D::j`, `Image< T >::resize()`, and `Image< T >::setVal()`.

Referenced by `Image< T >::countParticles()`, `Image< T >::drawDiskCheckTarget()`, `VisualBuffer::findMostInterestingTarget()`, `segmentLandmark()`, and `segmentObject()`.

#### 10.122.4.35 `template<class T> void Image< T >::freeMem () [inline]`

Free memory and switch to uninitialized state.

Note that it is **NOT** necessary to call this function to ensure proper cleanup, that will be done in the destructor by default. Rather, `freeMem()` is offered just as a performance optimization, to allow you to release a potentially large chunk of memory when you are finished using it.

Reimplemented in `ImageSpring< T >`, and `ImageSpring< Jet< float > >`.

Definition at line 871 of file `Image.H`.

References `Image< T >::swap()`.

Referenced by `ImageSpring< T >::freeMem()`, `segmentObjectClean()`, and `Jet< T >::~~Jet()`.

#### 10.122.4.36 `template<class T> T * Image< T >::getArrayPtr () [inline]`

Returns read/write (non-const) pointer to internal image array.

Definition at line 1075 of file `Image.H`.



#### 10.122.4.37 `template<class T> const T * Image< T >::getArrayPtr () const [inline]`

Returns read-only (const) pointer to internal image array.

Definition at line 1068 of file Image.H.

Referenced by `TCPmessage::addImage()`, `PsychoDisplay::displayText()`, `XWindow::drawImage()`, `fromRGB()`, `TCPmessage::getElement()`, `ImageSpring< T >::ImageSpring()`, `main()`, `Hmax::origGetC2()`, and `MPEGStream::WriteFrame()`.

#### 10.122.4.38 `template<class T> const Dims & Image< T >::getDims () const [inline]`

Get image width+height in [Dims](#) struct.

Definition at line 939 of file Image.H.

Referenced by `abs()`, `absDiff()`, `Image< T >::addDownsized()`, `Image< T >::addWeighted()`, `Image< T >::attenuateBorders()`, `average()`, `avgOrient()`, `binaryReverse()`, `centerSurroundDiff()`, `centerSurroundDiffSingleOpponent()`, `chamfer34()`, `clampedDiff()`, `Image< T >::clear()`, `colorStain()`, `ShapeEstimator::compute()`, `contour2D()`, `contour2D8()`, `MbariResultViewer::displayImage()`, `distDegrade()`, `RetinaStd::doGetOutput()`, `DirectFeedChannel::doInput()`, `downSizeClean()`, `BitObject::drawBoundingBox()`, `BitObject::drawOutline()`, `BitObject::drawShape()`, `VisualEventSet::drawTokens()`, `exp()`, `extractBitObjects()`, `SimulationViewerEyeMvt2::eyeMovement()`, `SimulationViewerEyeMvt::eyeMovement()`, `VisualBuffer::findMostInterestingTarget()`, `Image< T >::flood()`, `floodClean()`, `floodCleanBB()`, `SoxChannel::getNonlinearResponse()`, `getSalRegions()`, `Image< T >::getSize()`, `SingleChannelSurprise::getSurpriseMap()`, `SimulationViewerCompress::getTraj()`, `hmaxActivation()`, `infoFFT()`, `VisualBuffer::input()`, `SaliencyMapFast::input()`, `SaliencyMapStd::input()`, `inverse()`, `junctionFilterFull()`, `junctionFilterPartial()`, `SingleChannel::learn()`, `main()`, `makeBinary()`, `makeRGB()`, `negexp()`, `SurpriseImage< T >::neighborhoods()`, `normalizeRGPolar()`, `Brain::obtainOutput()`, `Image< T >::operator=()`, `overlay()`, `overlayStain()`, `remapRange()`, `replaceVals()`, `BitObject::reset()`, `Image< T >::resize()`, `BitObject::setMaxMinAvgIntensity()`, `Image< T >::splitPosNeg()`, `sqrt()`, `squared()`, `stain()`, `SurpriseMap< T >::surprise()`, `takeMax()`, `takeMin()`, `toPower()`, `toRGB()`, `VisualEventSet::updateEvents()`, and `writeImageToStream()`.

#### 10.122.4.39 `template<class T> int Image< T >::getHeight () const [inline]`

Get image height.

Reimplemented in [SaliencyMapStd](#), [SaliencyMapTrivial](#), [SaliencyMapFast](#), [TaskRelevanceMapStd](#), [TaskRelevanceMapKillStatic](#), [TaskRelevanceMapKillN](#), [Winner](#)

TakeAllStd, [WinnerTakeAllFast](#), and [WinnerTakeAllDummy](#).

Definition at line 934 of file Image.H.

Referenced by `Image< T >::addBGnoise()`, `Image< T >::addBGnoise2()`, `TCPmessage::addImage()`, `Image< T >::addScaled()`, `buildPyrGabor()`, `chamfer34()`, `SurpriseModelSP::combineFrom()`, `SurpriseModelSG::combineFrom()`, `computeCMAP()`, `ImageSpring< T >::computePos()`, `InferoTemporal::computeVisualObject()`, `contour2D()`, `contour2D8()`, `convolveHmax()`, `CINNIC::convolveTest()`, `Image< T >::coordsOk()`, `Image< T >::countParticles()`, `dct()`, `dilateImg()`, `ImageDisplay::display()`, `PsychoDisplay::displayText()`, `distDegrade()`, `RetinaStd::doGetOutput()`, `PN03contrastChannel::doInput()`, `ScorrChannel::doInput()`, `VarianceChannel::doInput()`, `EntropyChannel::doInput()`, `downSize()`, `downSizeClean()`, `Image< T >::drawArrow()`, `Image< T >::drawGrid()`, `XWindow::drawImage()`, `Image< T >::drawPoint()`, `Image< T >::embed()`, `erodeImg()`, `Brain::evolve()`, `SimulationViewerEyeMvt::eyeMovement()`, `featurePoolHmax()`, `Image< T >::findMax()`, `Image< T >::findMin()`, `VisualBuffer::findMostInterestingTargetLocMax()`, `fixationMask()`, `floodCleanBB()`, `ImageSpring< T >::getClusteredImage()`, `WinnerTakeAllDummy::getHeight()`, `WinnerTakeAllFast::getHeight()`, `WinnerTakeAllStd::getHeight()`, `TaskRelevanceMapKillN::getHeight()`, `TaskRelevanceMapKillStatic::getHeight()`, `TaskRelevanceMapStd::getHeight()`, `SaliencyMapFast::getHeight()`, `SaliencyMapTrivial::getHeight()`, `SaliencyMapStd::getHeight()`, `TcorrChannel::getOutput()`, `ImageSpring< T >::getPositions()`, `SimulationViewerStd::getTraj()`, `SimulationViewerEyeMvt2::getTraj()`, `SimulationViewerCompress::getTraj()`, `Image< T >::getValInterp()`, `Image< T >::getVector()`, `Image< T >::getVectorColumn()`, `Image2mexArray()`, `ImageSpring< T >::ImageSpring()`, `infoFFT()`, `ImageSpring< T >::initClustering()`, `Image< T >::initialized()`, `Image< T >::is1D()`, `Image< T >::isSameSize()`, `junctionFilterFull()`, `junctionFilterPartial()`, `lowPassPixel()`, `main()`, `matrixMult()`, `maxNormalizeFancy()`, `maxNormalizeFancyFast()`, `maxNormalizeFancyLandmark()`, `maxNormalizeFancyWeak()`, `maxNormalizeStd()`, `SurpriseImage< T >::neighborhoods()`, `normalizeWithScale()`, `omniDenebulize()`, `Image< T >::operator*()`, `Image< T >::operator+()`, `Image< T >::operator-()`, `Image< T >::operator/()`, `Image< T >::operator=()`, `optConvolve()`, `Image< T >::paste()`, `Image< T >::pasteGabor()`, `pasteImage()`, `Image< T >::rectangleOk()`, `FrameSeries::resizeImage()`, `SimulationViewerStd::saveResults()`, `Image< T >::setBorders()`, `spatialPoolMax()`, `Image< T >::speckleNoise()`, `sumXY()`, `SurpriseMap< T >::surprise()`, `transpose()`, `VisualEventSet::updateEvents()`, `writeBinImage()`, `MPEGStream::WriteFrame()`, and `Image< T >::writeText()`.

#### 10.122.4.40 `template<class T> double Image< T >::getMean \(\) const`

compute mean pixel value over image (promoted to double)

Reimplemented in [SurpriseImage< T >](#), and [SurpriseImage< SurpriseModelSP >](#).

Definition at line 145 of file Image\_MonoOnly.C.

References ASSERT, Image< T >::end(), Image< T >::getSize(), and Image< T >::initialized().

Referenced by SurpriseImage< T >::getMean(), and Image< T >::getStdev().

#### 10.122.4.41 `template<class T> void Image< T >::getMinMax (T & mini, T & maxi) const [inline]`

Get min and max values.

Definition at line 954 of file Image.H.

References ASSERT, Image< T >::begin(), Image< T >::const\_iterator, Image< T >::end(), and Image< T >::initialized().

Referenced by ComplexChannel::combineOutputs(), SingleChannel::combineSubMaps(), TaskRelevanceMapKillStatic::evolve(), VisualBuffer::findMostInterestingTarget(), VisualBuffer::findMostInterestingTargetLocMax(), SingleChannel::getRawCMap(), infoFFT(), TaskRelevanceMapKillN::input(), main(), maxNormalize(), maxNormalizeFancy(), maxNormalizeFancyFast(), maxNormalizeFancyLandmark(), maxNormalizeFancyWeak(), Image< T >::normalize(), SingleChannel::postProcessMap(), and squash().

#### 10.122.4.42 `template<class T> void Image< T >::getMinMaxAvg (T & mini, T & maxi, T & avg) const`

Get min, max and average values.

Definition at line 81 of file Image\_MonoOnly.C.

References ASSERT, Image< T >::begin(), Image< T >::const\_iterator, Image< T >::end(), Image< T >::getSize(), and Image< T >::initialized().

Referenced by SimulationViewerEyeMvt2::eyeMovement(), SimulationViewerEyeMvt::eyeMovement(), and SimulationViewerCompress::getTraj().

#### 10.122.4.43 `template<class T> T Image< T >::getSample (const Point2D & center, const int radius)`

Get a sample which is the max value within a circular aperture.

This only works for monochromatic images.

##### Parameters:

**center** coordinates of the center of the aperture

**radius** radius of the aperture, in pixels

Definition at line 60 of file Image\_MonoOnly.C.

References ASSERT, Image< T >::coordsOk(), Image< T >::getVal(), Point2D::i, Image< T >::initialized(), and Point2D::j.

Referenced by SimulationViewerEyeMvt2::eyeMovement(), and SimulationViewerEyeMvt::eyeMovement().

#### 10.122.4.44 **template<class T> int Image< T >::getSize () const** [inline]

Get image size (width \* height).

Definition at line 924 of file Image.H.

References Image< T >::getDims(), and Dims::sz().

Referenced by TCPmessage::addImage(), avgOrient(), chamfer34(), ImageSpring< T >::computePos(), ImageSpring< T >::computeStiff(), corrcoeff(), Image< T >::flood(), ImageSpring< T >::freeMem(), Image< T >::getMean(), Image< T >::getMinMaxAvg(), ImageSpring< T >::getPositions(), Image< T >::getStdev(), Image< T >::getVal(), hmaxActivation(), Image2mexArray(), ImageSpring< T >::init(), learningCoeff(), main(), mean(), Image< T >::operator[](), pSNR(), Image< T >::retainMax(), RMSerr(), segmentLandmark(), segmentObject(), segmentObjectClean(), and Image< T >::setVal().

#### 10.122.4.45 **template<class T> double Image< T >::getStdev () const**

compute standard deviation of pixel values over image

Definition at line 154 of file Image\_MonoOnly.C.

References ASSERT, Image< T >::begin(), Image< T >::const\_iterator, Image< T >::end(), Image< T >::getMean(), Image< T >::getSize(), Image< T >::initialized(), and sqrt().

Referenced by Image< T >::varianceNormalize().

#### 10.122.4.46 **template<class T> template<class T2> void Image< T >::getVal (const int x, const int y, T2 & val) const** [inline]

Get value at (x,y), put in val rather than return it.

Definition at line 1009 of file Image.H.

References ASSERT, Image< T >::begin(), Image< T >::coordsOk(), and Image< T >::getWidth().

**10.122.4.47** `template<class T> const T & Image< T >::getVal (const Point2D & p) const [inline]`

Get pixel value at specified coordinates in image.

Definition at line 1001 of file Image.H.

References ASSERT, Image< T >::begin(), Image< T >::coordsOk(), Image< T >::getWidth(), Point2D::i, and Point2D::j.

**10.122.4.48** `template<class T> const T & Image< T >::getVal (const int x, const int y) const [inline]`

Get pixel value at (x, y) in image.

Definition at line 993 of file Image.H.

References ASSERT, Image< T >::begin(), Image< T >::coordsOk(), and Image< T >::getWidth().

**10.122.4.49** `template<class T> const T & Image< T >::getVal (const int index) const [inline]`

Get pixel value at index in image.

Reimplemented in [Jet< T >](#), and [Jet< float >](#).

Definition at line 985 of file Image.H.

References ASSERT, Image< T >::begin(), and Image< T >::getSize().

Referenced by ShapeEstimator::compute(), ImageSpring< T >::computeStiff(), CINNIC::convolveTest(), copy(), Image< T >::countParticles(), VisualObject::createKeys(), dct(), SaliencyMapStd::depress(), PN03contrastChannel::doInput(), BitObject::drawBoundingBox(), Image< T >::drawDiskCheckTarget(), SimulationViewerStd::drawMaskOutline(), BitObject::drawOutline(), extractBitObjects(), SimulationViewerEyeMvt2::eyeMovement(), VisualBuffer::findMostInterestingTargetLocMax(), Image< T >::flood(), floodClean(), floodCleanBB(), ImageSpring< T >::getClusteredImage(), Image< T >::getSample(), ImageSpring< T >::getStats(), ImageSpring< T >::getStatsDist(), SimulationViewerEyeMvt2::getTraj(), SimulationViewerCompress::getTraj(), Jet< T >::getVal(), Jet< T >::getValV(), infoFFT(), lowPassPixel(), main(), matrixMult(), maxNormalizeLandmark(), maxNormalizeStd(), omniDenebulize(), segmentLandmark(), segmentObject(), segmentObjectClean(), SurpriseMap< T >::surprise(), TaskRelevanceMapStd::update(), and warp3D().

**10.122.4.50** `template<class T> const T Image< T >::getValInterp (const float x, const float y) const [inline]`

Get pixel value at (x, y) in image with bilinear interpolation.

Definition at line 1017 of file Image.H.

References ASSERT, Image< T >::begin(), Image< T >::const\_iterator, Image< T >::coordsOk(), Image< T >::getHeight(), and Image< T >::getWidth().

**10.122.4.51** `template<class T> void Image< T >::getVector (std::vector< std::vector< T * > > * V) const [inline]`

return a double nested vector that represents data (see also =operator)

Definition at line 1352 of file Image.H.

References ASSERT, Image< T >::begin(), Image< T >::const\_iterator, Image< T >::getHeight(), Image< T >::getWidth(), and Image< T >::iterator.

**10.122.4.52** `template<class T> void Image< T >::getVector (std::vector< std::vector< T > > * V) const [inline]`

return a double nested vector that represents data (see also =operator)

Definition at line 1334 of file Image.H.

References ASSERT, Image< T >::begin(), Image< T >::const\_iterator, Image< T >::getHeight(), Image< T >::getWidth(), and Image< T >::iterator.

**10.122.4.53** `template<class T> void Image< T >::getVectorColumn (std::vector< T * > * V, int column) const [inline]`

return a single column of an image as a vector

Definition at line 1416 of file Image.H.

References ASSERT, Image< T >::begin(), Image< T >::getHeight(), Image< T >::getWidth(), and Image< T >::iterator.

**10.122.4.54** `template<class T> void Image< T >::getVectorColumn (std::vector< T > * V, int column) const [inline]`

return a single column of an image as a vector

Definition at line 1401 of file Image.H.

References ASSERT, Image< T >::begin(), Image< T >::getHeight(), Image< T >::getWidth(), and Image< T >::iterator.

**10.122.4.55** `template<class T> void Image< T >::getVectorRow (std::vector< T * > * V, int row) const [inline]`

return a single row of an image as a vector

Definition at line 1385 of file Image.H.

References ASSERT, Image< T >::begin(), Image< T >::const\_iterator, Image< T >::getWidth(), and Image< T >::iterator.

**10.122.4.56** `template<class T> void Image< T >::getVectorRow (std::vector< T > * V, int row) const [inline]`

return a single row of an image as a vector

Definition at line 1370 of file Image.H.

References ASSERT, Image< T >::begin(), Image< T >::const\_iterator, Image< T >::getWidth(), and Image< T >::iterator.

**10.122.4.57** `template<class T> int Image< T >::getWidth () const [inline]`

Get image width.

Reimplemented in [SaliencyMapStd](#), [SaliencyMapTrivial](#), [SaliencyMapFast](#), [TaskRelevanceMapStd](#), [TaskRelevanceMapKillStatic](#), [TaskRelevanceMapKillN](#), [WinnerTakeAllStd](#), [WinnerTakeAllFast](#), and [WinnerTakeAllDummy](#).

Definition at line 929 of file Image.H.

Referenced by Image< T >::addBGnoise(), Image< T >::addBGnoise2(), TCPmessage::addImage(), Image< T >::addScaled(), buildPyrGabor(), chamfer34(), SurpriseModelSP::combineFrom(), SurpriseModelSG::combineFrom(), computeCMAP(), ImageSpring< T >::computePos(), IneroTemporal::computeVisualObject(), contour2D(), contour2D8(), convolveHmax(), CINNIC::convolveTest(), Image< T >::coordsOk(), corrpach(), Image< T >::countParticles(), dct(), dilateImg(), ImageDisplay::display(), PsychoDisplay::displayText(), distDegrade(), RetinaStd::doGetOutput(), PN03contrastChannel::doInput(), ScorrChannel::doInput(), VarianceChannel::doInput(), EntropyChannel::doInput(), downSize(), downSizeClean(), Image< T >::drawArrow(), Image< T >::drawGrid(), XWindow::drawImage(), Image< T >::drawPoint(), BitObject::drawShape(), Image< T >::embed(), erodeImg(), Brain::evolve(), SimulationViewerEyeMvt::eyeMovement(), featurePoolHmax(), Image< T >::findMax(), Image< T >::findMin(), VisualBuffer::findMost-

InterestingTargetLocMax(), fixationMask(), Image< T >::flood(), floodCleanBB(), LPTFoveator::foveate(), ImageSpring< T >::getClusteredImage(), ImageSpring< T >::getIndex(), SoxChannel::getNonlinearResponse(), TcorrChannel::getOutput(), SimulationViewerStd::getTraj(), SimulationViewerEyeMvt2::getTraj(), SimulationViewerEyeMvt::getTraj(), SimulationViewerCompress::getTraj(), Image< T >::getVal(), Image< T >::getValInterp(), Image< T >::getVector(), Image< T >::getVectorColumn(), Image< T >::getVectorRow(), WinnerTakeAllDummy::getWidth(), WinnerTakeAllFast::getWidth(), WinnerTakeAllStd::getWidth(), TaskRelevanceMapKillN::getWidth(), TaskRelevanceMapKillStatic::getWidth(), TaskRelevanceMapStd::getWidth(), SaliencyMapFast::getWidth(), SaliencyMapTrivial::getWidth(), SaliencyMapStd::getWidth(), ImageSpring< T >::getXY(), Beobot::highLevel(), Image2mexArray(), ImageSpring< T >::ImageSpring(), infoFFT(), ImageSpring< T >::initClustering(), Image< T >::initialized(), Image< T >::is1D(), Image< T >::isSameSize(), junctionFilterFull(), junctionFilterPartial(), lowPassPixel(), main(), matrixMult(), maxNormalizeFancy(), maxNormalizeFancyFast(), maxNormalizeFancyLandmark(), maxNormalizeFancyWeak(), maxNormalizeStd(), SurpriseImage< T >::neighborhoods(), normalizeWithScale(), omniDenebulize(), Image< T >::operator \*(), Image< T >::operator +(), Image< T >::operator -(), Image< T >::operator /(), Image< T >::operator =(), optConvolve(), Image< T >::paste(), Image< T >::pasteGabor(), pasteImage(), SingleChannel::postProcessMap(), Image< T >::rectangleOk(), FrameSeries::resizeImage(), SimulationViewerStd::saveResults(), Image< T >::setBorders(), BitObject::setMaxMinAvgIntensity(), Image< T >::setVal(), spatialPoolMax(), Image< T >::speckleNoise(), sumXY(), SurpriseMap< T >::surprise(), transpose(), VisualEventSet::updateEvents(), writeBinImage(), MPEGStream::WriteFrame(), and Image< T >::writeText().

#### 10.122.4.58 `template<class T> bool Image< T >::hasSameData (const Image< T > & b) const [inline]`

For testing/debugging only.

See if we are pointing to the same ArrayData<T> as is the other Image<T>.

Definition at line 1434 of file Image.H.

References Image< T >::itsHdl.

#### 10.122.4.59 `template<class T> bool Image< T >::initialized () const [inline]`

Check whether image is non-empty (i.e., non-zero height and width).

Definition at line 835 of file Image.H.

References Image< T >::getHeight(), and Image< T >::getWidth().

Referenced by Image< T >::addBGnoise(), Image< T >::addBGnoise2(), Image< T



>::addScaled(), Image< T >::addWeighted(), DirectFeedChannel::applyMaxNorm(), SimulationViewerStd::attentionShift(), SimulationViewerEyeMvt2::attentionShift(), Image< T >::attenuateBorders(), chamfer34(), contour2D(), contour2D8(), convolveHmax(), copy(), Image< T >::countThresh(), dct(), dilateImg(), BitObject::drawBoundingBox(), Image< T >::drawCircle(), Image< T >::drawCross(), Image< T >::drawDisk(), Image< T >::drawDiskCheckTarget(), Image< T >::drawGrid(), Image< T >::drawLine(), BitObject::drawOutline(), Image< T >::drawPatch(), Image< T >::drawRect(), BitObject::drawShape(), Image< T >::embed(), erodeImg(), SimulationViewerStd::eyeMovement(), Image< T >::findMax(), Image< T >::findMin(), Image< T >::flood(), floodClean(), floodCleanBB(), Image< T >::getMean(), Image< T >::getMinMax(), Image< T >::getMinMaxAvg(), Image< T >::getSample(), Image< T >::getStddev(), goodness\_map(), SimulationViewerStd::headMovement(), highThresh(), ImageSpring< T >::ImageSpring(), VisualBuffer::input(), Image< T >::isFinite(), learningCoeff(), Image< T >::lowThresh(), Image< T >::lowThreshAbs(), main(), matrixMult(), maxNormalizeFancy(), maxNormalizeFancyFast(), maxNormalizeFancyLandmark(), maxNormalizeFancyWeak(), maxNormalizeLandmark(), maxNormalizeStd(), Image< T >::normalize(), optConvolve(), Brain::processInput(), Image< T >::rectify(), Image< T >::replaceAll(), BitObject::reset(), RMSerr(), segmentLandmark(), segmentObject(), segmentObjectClean(), Image< T >::setBorders(), SingleChannel::setClipPyramid(), Image< T >::setValMask(), showAllObjects(), Image< T >::sigmoid(), Image< T >::speckleNoise(), Image< T >::splitPosNeg(), Image< T >::square(), toRGB(), Image< T >::varianceNormalize(), and Image< T >::wilsonNormalize().

#### 10.122.4.60 `template<class T> bool Image< T >::is1D () const` [inline]

Check if the image is 1D, i.e., width == 1 or height == 1.

Definition at line 949 of file Image.H.

References Image< T >::getHeight(), and Image< T >::getWidth().

#### 10.122.4.61 `template<class T> bool Image< T >::isFinite () const`

Check whether all pixels have finite values.

This relies on `isFinite()` being defined for your pixel type T. In [saliency.H](#) we define it for the canonical types, and you can use these canonical definitions to define it for more complex types (see, for example, the definition in [Pixels.H](#)).

Definition at line 99 of file Image\_MonoOnly.C.

References ASSERT, Image< T >::begin(), Image< T >::const\_iterator, Image< T >::end(), and Image< T >::initialized().

#### 10.122.4.62 `template<class T> template<class C> bool Image< T >::isSameSize (const C & other) const [inline]`

Check if \*this is the same size as the other thing.

The other thing can be any type that exposes [getHeight\(\)](#) and [getWidth\(\)](#)

Definition at line 944 of file Image.H.

References `Image< T >::getHeight()`, and `Image< T >::getWidth()`.

Referenced by `absDiff()`, `Image< T >::addScaled()`, `SimulationViewerStd::attentionShift()`, `average()`, `averageWeighted()`, `clampedDiff()`, `SurpriseModelSP::combineFrom()`, `SurpriseModelSG::combineFrom()`, `corrcoef()`, `distance()`, `doubleOpp()`, `SimulationViewerStd::eyeMovement()`, `featurePoolHmax()`, `SimulationViewerStd::headMovement()`, `SurpriseImage< T >::init()`, `junctionFilterFull()`, `junctionFilterPartial()`, `learningCoeff()`, `makeRGB()`, `Brain::obtainOutput()`, `Image< T >::operator*()`, `Image< T >::operator*=(())`, `Image< T >::operator+()`, `Image< T >::operator+=(())`, `Image< T >::operator-()`, `Image< T >::operator-=(())`, `Image< T >::operator/()`, `Image< T >::operator/=(())`, `Jet< T >::operator=(())`, `overlay()`, `overlayStain()`, `RMSerr()`, `Image< T >::setValMask()`, `takeMax()`, and `takeMin()`.

#### 10.122.4.63 `template<class T> void Image< T >::lowThresh (const T thresh, const T val = T())`

Cut values < thresh and replace them by val.

Definition at line 276 of file Image\_MonoOnly.C.

References `ASSERT`, `Image< T >::beginw()`, `Image< T >::endw()`, `Image< T >::initialized()`, and `Image< T >::iterator`.

Referenced by `SimulationViewerStd::drawMaskOutline()`, `SimulationViewerEyeMvt2::eyeMovement()`, `SimulationViewerCompress::getTraj()`, `VisualBuffer::input()`, and `main()`.

#### 10.122.4.64 `template<class T> void Image< T >::lowThreshAbs (const T thresh, const T val = T())`

Cut values whose abs is < thresh and replace them by val.

Definition at line 290 of file Image\_MonoOnly.C.

References `abs()`, `ASSERT`, `Image< T >::beginw()`, `Image< T >::endw()`, `Image< T >::initialized()`, and `Image< T >::iterator`.

#### 10.122.4.65 `template<class T> void Image< T >::luminanceNormalize (const Image< T > & lum, const T thresh)`

Normalize my values by those in the lum image.

Definition at line 229 of file Image\_MonoOnly.C.

References ASSERT, Image< T >::beginw(), Image< T >::const\_iterator, Image< T >::endw(), Image< T >::iterator, and lum.

#### 10.122.4.66 `template<class T> void Image< T >::normalize (const T nmin, const T nmax)`

Normalize values between nmin and nmax.

Definition at line 180 of file Image\_MonoOnly.C.

References ASSERT, Image< T >::beginw(), Image< T >::endw(), Image< T >::getMinMax(), Image< T >::initialized(), and Image< T >::iterator.

Referenced by contrastModulate(), SimulationViewerStd::drawMegaCombo(), SimulationViewerEyeMvt2::eyeMovement(), VisualBuffer::getBuffer(), SimulationViewerStd::getTraj(), WinnerTakeAllDummy::getV(), WinnerTakeAllFast::getV(), WinnerTakeAllStd::getV(), SaliencyMapFast::getV(), SaliencyMapTrivial::getV(), SaliencyMapStd::getV(), AttentionGuidanceMapStd::getV(), goodness\_map(), main(), maxNormalizeFancy(), maxNormalizeFancyFast(), maxNormalizeFancyLandmark(), maxNormalizeFancyWeak(), maxNormalizeLandmark(), maxNormalizeNone(), maxNormalizeStd(), normalizeRGB(), normalizeWithScale(), Image< T >::varianceNormalize(), Raster::WriteFloat(), and Raster::WriteFloatNorm().

#### 10.122.4.67 `template<class T> template<class T2> Image< typename promote_trait< T, T2 >::TP > Image< T >::operator * (const Image< T2 > & img) const [inline]`

Point-wise multiply \*this by img and return result.

Definition at line 1276 of file Image.H.

References ASSERT, Image< T >::begin(), Image< T >::beginw(), Image< T >::const\_iterator, Image< T >::end(), Image< T >::getHeight(), Image< T >::getWidth(), Image< T >::isSameSize(), Image< T >::iterator, and NO\_INIT.

#### 10.122.4.68 `template<class T> template<class T2> Image< typename promote_trait< T, T2 >::TP > Image< T >::operator * (const T2 & val) const [inline]`

Multiply scalar each point in \*this by scalar and return result.

Definition at line 1220 of file Image.H.

References Image< T >::begin(), Image< T >::beginw(), Image< T >::const\_iterator, Image< T >::end(), Image< T >::getHeight(), Image< T >::getWidth(), Image< T >::iterator, and NO\_INIT.

**10.122.4.69** `template<class T> template<class T2> Image< T > & Image< T >::operator *= (const Image< T2 > & A) [inline]`

Multiply image by image, point-by-point, clamp result as necessary.

Definition at line 1170 of file Image.H.

References ASSERT, Image< T >::begin(), Image< T >::beginw(), Image< T >::const\_iterator, Image< T >::endw(), Image< T >::isSameSize(), and Image< T >::iterator.

**10.122.4.70** `template<class T> Image< T > & Image< T >::operator *= (const T & val) [inline]`

Multiply image by constant, clamp result as necessary.

Definition at line 1126 of file Image.H.

References Image< T >::beginw(), Image< T >::endw(), and Image< T >::iterator.

**10.122.4.71** `template<class T> template<class T2> Image< typename promote_trait< T, T2 >::TP > Image< T >::operator+ (const Image< T2 > & img) const [inline]`

Point-wise add img to image and return result.

Definition at line 1246 of file Image.H.

References ASSERT, Image< T >::begin(), Image< T >::beginw(), Image< T >::const\_iterator, Image< T >::end(), Image< T >::getHeight(), Image< T >::getWidth(), Image< T >::isSameSize(), Image< T >::iterator, and NO\_INIT.

**10.122.4.72** `template<class T> template<class T2> Image< typename promote_trait< T, T2 >::TP > Image< T >::operator+ (const T2 & val) const [inline]`

Add scalar to each point in \*this and return result.

Definition at line 1194 of file Image.H.

References Image< T >::begin(), Image< T >::beginw(), Image< T >::const\_iterator, Image< T >::end(), Image< T >::getHeight(), Image< T >::getWidth(),

Image< T >::iterator, and NO\_INIT.

**10.122.4.73** `template<class T> template<class T2> Image< T > & Image< T >::operator+=(const Image< T2 > & A) [inline]`

Add image to image, clamp result as necessary.

Definition at line 1146 of file Image.H.

References ASSERT, Image< T >::begin(), Image< T >::beginw(), Image< T >::const\_iterator, Image< T >::endw(), Image< T >::isSameSize(), and Image< T >::iterator.

**10.122.4.74** `template<class T> Image< T > & Image< T >::operator+=(const T & val) [inline]`

Add constant to image, clamp result as necessary.

Definition at line 1106 of file Image.H.

References Image< T >::beginw(), Image< T >::endw(), and Image< T >::iterator.

**10.122.4.75** `template<class T> template<class T2> Image< typename promote_trait< T, T2 >::TP > Image< T >::operator-(const Image< T2 > & img) const [inline]`

Point-wise subtract img from \*this and return result.

Definition at line 1261 of file Image.H.

References ASSERT, Image< T >::begin(), Image< T >::beginw(), Image< T >::const\_iterator, Image< T >::end(), Image< T >::getHeight(), Image< T >::getWidth(), Image< T >::isSameSize(), Image< T >::iterator, and NO\_INIT.

**10.122.4.76** `template<class T> template<class T2> Image< typename promote_trait< T, T2 >::TP > Image< T >::operator-(const T2 & val) const [inline]`

Subtract scalar from each point in \*this and return result.

Definition at line 1207 of file Image.H.

References Image< T >::begin(), Image< T >::beginw(), Image< T >::const\_iterator, Image< T >::end(), Image< T >::getHeight(), Image< T >::getWidth(), Image< T >::iterator, and NO\_INIT.

**10.122.4.77** `template<class T> template<class T2> Image< T > & Image< T >::operator-= (const Image< T2 > & A) [inline]`

Subtract image from image, clamp result as necessary.

Definition at line 1158 of file Image.H.

References ASSERT, Image< T >::begin(), Image< T >::beginw(), Image< T >::const\_iterator, Image< T >::endw(), Image< T >::isSameSize(), and Image< T >::iterator.

**10.122.4.78** `template<class T> Image< T > & Image< T >::operator-= (const T & val) [inline]`

Subtract constant from image, clamp result as necessary.

Definition at line 1116 of file Image.H.

References Image< T >::beginw(), Image< T >::endw(), and Image< T >::iterator.

**10.122.4.79** `template<class T> template<class T2> Image< typename promote_trait< T, T2 >::TP > Image< T >::operator/ (const Image< T2 > & img) const [inline]`

Point-wise divide \*this by img and return result.

Definition at line 1291 of file Image.H.

References ASSERT, Image< T >::begin(), Image< T >::beginw(), Image< T >::const\_iterator, Image< T >::end(), Image< T >::getHeight(), Image< T >::getWidth(), Image< T >::isSameSize(), Image< T >::iterator, and NO\_INIT.

**10.122.4.80** `template<class T> template<class T2> Image< typename promote_trait< T, T2 >::TP > Image< T >::operator/ (const T2 & val) const [inline]`

Divide each point in \*this by scalar and return result.

Definition at line 1233 of file Image.H.

References Image< T >::begin(), Image< T >::beginw(), Image< T >::const\_iterator, Image< T >::end(), Image< T >::getHeight(), Image< T >::getWidth(), Image< T >::iterator, and NO\_INIT.

**10.122.4.81** `template<class T> template<class T2> Image< T > & Image< T >::operator/= (const Image< T2 > & A) [inline]`

Divide image by image, point-by-point, clamp result as necessary.

Definition at line 1182 of file Image.H.

References ASSERT, Image< T >::begin(), Image< T >::beginw(), Image< T >::const\_iterator, Image< T >::endw(), Image< T >::isSameSize(), and Image< T >::iterator.

**10.122.4.82** `template<class T> Image< T > & Image< T >::operator/= (const T & val) [inline]`

Divide image by constant, clamp result as necessary.

Definition at line 1136 of file Image.H.

References Image< T >::beginw(), Image< T >::endw(), and Image< T >::iterator.

**10.122.4.83** `template<class T> template<class T2> Image< T > & Image< T >::operator= (const Image< T2 > & A) [inline]`

Conversion assignment operator.

e.g.:

```
Image<byte> im1; Image<float> im2; im2 = im1;
```

Definition at line 796 of file Image.H.

References Image< T >::begin(), Image< T >::beginw(), Image< T >::const\_iterator, Image< T >::endw(), Image< T >::getDims(), Image< T >::iterator, and NO\_INIT.

**10.122.4.84** `template<class T> Image< T > & Image< T >::operator= (std::vector< std::vector< T > > & A) [inline]`

Conversion assignment operator.

e.g.:

```
Image<byte> im1; std::vector im2; im1 = im2;
```

Definition at line 734 of file Image.H.

References Image< T >::beginw(), Image< T >::getHeight(), Image< T >::getWidth(), Image< T >::iterator, and Image< T >::resize().

#### 10.122.4.85 `template<class T> Image< T > & Image< T >::operator= (const Image< T > & A) [inline]`

Assignment operator.

e.g.:

```
Image<byte> im1, im2; im2 = im1;
```

Reimplemented in [Jet< T >](#), and [Jet< float >](#).

Definition at line 725 of file [Image.H](#).

References [Image< T >::swap\(\)](#).

Referenced by [Jet< T >::operator=\(\)](#).

#### 10.122.4.86 `]`

`template<class T> const T & Image< T >::operator[] (const int index) const [inline]`

Access image elements through C array index interface.

Definition at line 977 of file [Image.H](#).

References [ASSERT](#), [Image< T >::begin\(\)](#), and [Image< T >::getSize\(\)](#).

#### 10.122.4.87 `]`

`template<class T> T & Image< T >::operator[] (const int index) [inline]`

Access image elements through C array index interface.

Definition at line 969 of file [Image.H](#).

References [ASSERT](#), [Image< T >::beginw\(\)](#), and [Image< T >::getSize\(\)](#).

#### 10.122.4.88 `template<class T> void Image< T >::paste (const Image< T > & img, const Point2D & pos)`

Paste image into \*this at given position.

Image must fit within \*this and no resizing/interpolation is done. See [embed\(\)](#) for a more sophisticated version with interpolation and resizing.

Definition at line 292 of file [Image.C](#).

References [ASSERT](#), [Image< T >::begin\(\)](#), [Image< T >::beginw\(\)](#), [Image< T >::const\\_iterator](#), [Image< T >::getHeight\(\)](#), [Image< T >::getWidth\(\)](#), [Point2D::i](#), [Image< T >::iterator](#), and [Point2D::j](#).



Referenced by RetinaStd::doGetOutput(), SimulationViewerEyeMvt2::getTraj(), SimulationViewerCompress::getTraj(), main(), normalizeWithScale(), and CINNIC::viewNeuronTemplate().

#### 10.122.4.89 `template<class T> void Image< T >::pasteGabor (const Image< T > & gabor, const Point2D & pos, const T background)`

Paste a gabor into an image area. Account for overlap.

this is a simple extention to paste, but it tests for gabor overlap and attempts rectify similar to solution for MakeSnake by taking the pixel value furthest from the background value

##### Parameters:

*background* This is the color of the image background

Definition at line 308 of file Image.C.

References ASSERT, Image< T >::begin(), Image< T >::beginw(), Image< T >::const\_iterator, Image< T >::getHeight(), Image< T >::getWidth(), Point2D::i, Image< T >::iterator, and Point2D::j.

#### 10.122.4.90 `template<class T> bool Image< T >::rectangleOk (const Rectangle & rect) const [inline]`

Test whether rectangle fits in image.

Definition at line 1096 of file Image.H.

References Rectangle::bottom(), Image< T >::getHeight(), Image< T >::getWidth(), Rectangle::left(), Rectangle::right(), and Rectangle::top().

Referenced by copy(), corrpach(), Image< T >::drawRect(), and Image< T >::embed().

#### 10.122.4.91 `template<class T> void Image< T >::rectify ()`

Saturate values < 0.

Definition at line 246 of file Image\_MonoOnly.C.

References ASSERT, Image< T >::beginw(), Image< T >::endw(), Image< T >::initialized(), and Image< T >::iterator.

Referenced by DisparityChannel::doInput(), maxNormalizeFancy(), maxNormalizeFancyFast(), maxNormalizeFancyLandmark(), maxNormalizeFancyWeak(), maxNormalizeNone(), maxNormalizeStd(), and SurpriseMap< T >::surprise().

#### 10.122.4.92 **template<class T> long [Image](#)< T >::refCount () const throw ()** [inline]

For testing/debugging only.

Returns the current reference count.

Definition at line 1439 of file Image.H.

#### 10.122.4.93 **template<class T> void [Image](#)< T >::replaceVal (const T & *from*, const T & *to*)**

Replace all occurrences of a value 'from' by another value 'to'.

Definition at line 93 of file Image.C.

References ASSERT, [Image](#)< T >::beginw(), [Image](#)< T >::endw(), [Image](#)< T >::initialized(), and [Image](#)< T >::iterator.

Referenced by [VisualBuffer::findMostInterestingTarget\(\)](#).

#### 10.122.4.94 **template<class T> void [Image](#)< T >::resize (const int *width*, const int *height*, const bool *clear* = false) [inline]**

Free mem and realloc new array (array contents are lost).

Use [rescale\(\)](#) instead if you want to preserve image contents.

Definition at line 864 of file Image.H.

References [Image](#)< T >::resize().

#### 10.122.4.95 **template<class T> void [Image](#)< T >::resize (const [Dims](#) & *dims*, const bool *clear* = false) [inline]**

Free mem and realloc new array (array contents are lost).

Use [rescale\(\)](#) instead if you want to preserve image contents.

Definition at line 840 of file Image.H.

References [Image](#)< T >::clear(), and [Image](#)< T >::getDims().

Referenced by [SimulationViewerEyeMvt2::attentionShift\(\)](#), [CINNIC::convolveTest\(\)](#), [SimulationViewerStd::drawMegaCombo\(\)](#), [VisualEventSet::drawTokens\(\)](#), [contourRun::dumpEnergySigmoid\(\)](#), [Image](#)< T >::flood(), [floodClean\(\)](#), [floodCleanBB\(\)](#), [SimulationViewerEyeMvt2::getTraj\(\)](#), [Jet](#)< T >::init(), [ImageSpring](#)< T >::init(), [VisualBuffer::input\(\)](#), [Jet](#)< T >::Jet(), [main\(\)](#), [maxNormalizeLandmark\(\)](#), [Image](#)< T >::operator=(), [Hmax::origGetC2\(\)](#), [readImageFromStream\(\)](#), [Image](#)< T >::resize(),

Image< T >::splitPosNeg(), SurpriseMap< T >::surprise(), and CINNIC::view-NeuronTemplate().

#### 10.122.4.96 `template<class T> void Image< T >::retainMax (const int num)`

retain the num largest values and set the other entries to zero

Definition at line 364 of file Image\_MonoOnly.C.

References Image< T >::begin(), Image< T >::const\_iterator, Image< T >::getSize(), Image< T >::iterator, and Image< T >::setVal().

#### 10.122.4.97 `template<class T> void Image< T >::setBorders (const int size, const T value = T())`

set borders of "size" pixels to given value

(default value is T's default, which should be T's representation of zero).

Definition at line 254 of file Image.C.

References ASSERT, Image< T >::beginw(), Image< T >::getHeight(), Image< T >::getWidth(), Image< T >::initialized(), Image< T >::iterator, and safecopy().

#### 10.122.4.98 `template<class T> template<class T2> void Image< T >::setVal (const int x, const int y, const T2 & value) [inline]`

Set value in image at (x, y).

Definition at line 1052 of file Image.H.

References ASSERT, Image< T >::beginw(), Image< T >::coordsOk(), and Image< T >::getWidth().

#### 10.122.4.99 `template<class T> template<class T2> void Image< T >::setVal (const Point2D & p, const T2 & value) [inline]`

Set value in image at [Point2D](#).

Definition at line 1060 of file Image.H.

References ASSERT, Image< T >::beginw(), Image< T >::coordsOk(), Image< T >::getWidth(), [Point2D::i](#), and [Point2D::j](#).

#### 10.122.4.100 `template<class T> template<class T2> void Image< T >::setVal (const int index, const T2 & value) [inline]`

Set value in image at index.

Definition at line 1044 of file Image.H.

References ASSERT, Image< T >::beginw(), and Image< T >::getSize().

Referenced by binomialKernel(), CINNIC::convolveTest(), dct(), BitObject::drawBoundingBox(), Image< T >::drawCircle(), Image< T >::drawDisk(), Image< T >::drawDiskCheckTarget(), Image< T >::drawGrid(), BitObject::drawOutline(), Image< T >::drawPatch(), Image< T >::drawPoint(), contourRun::dumpEnergySigmoid(), VisualBuffer::findMostInterestingTarget(), Image< T >::flood(), floodClean(), floodCleanBB(), PyrFoveator::foveate(), LPTFoveator::foveate(), BlurFoveator::foveate(), gaussian(), gaussianBlob(), ImageSpring< T >::getClusteredImage(), VisualCortex::getJetDistMap(), LPTFoveator::getLPT(), VisualCortex::getRaoJetMap(), infoFFT(), infoMeasure(), LPTFoveator::invLPT(), lowPassPixel(), main(), omniCorrectGen(), omniCorrectSp(), Image< T >::retainMax(), BeobotVisualCortex::singleCPUprocess(), Image< T >::speckleNoise(), MultiConvolveChannel::start1(), warp3D(), and Image< T >::writeText().

#### 10.122.4.101 `template<class T> void Image< T >::setValMask (const Image< byte > & mask, const T val)`

Set values at all locations where mask is non-zero.

Definition at line 80 of file Image.C.

References ASSERT, Image< T >::begin(), Image< T >::beginw(), Image< T >::const\_iterator, Image< T >::endw(), Image< T >::initialized(), Image< T >::isSameSize(), and Image< T >::iterator.

#### 10.122.4.102 `template<class T> void Image< T >::sigmoid (const float s)`

Pass image through sigmoid:  $f(x) = x^2 / (s + x^2)$ .

Definition at line 349 of file Image\_MonoOnly.C.

References ASSERT, Image< T >::beginw(), Image< T >::endw(), Image< T >::initialized(), and Image< T >::iterator.

#### 10.122.4.103 `template<class T> void Image< T >::sigmoid (const float g, const float h, const float s)`

Pass image through sigmoid:  $f(x) = x^g / (s + x^h)$ .

Definition at line 336 of file Image\_MonoOnly.C.

References ASSERT, Image< T >::beginw(), Image< T >::endw(), Image< T >::initialized(), and Image< T >::iterator.

**10.122.4.104** `template<class T> void Image< T >::speckleNoise (const float thresh, const int size, const T noise_color, bool random_color = false)`

Add speckle noise to array; thresh = 1 for 100% noise.

Definition at line 623 of file Image.C.

References ASSERT, Image< T >::getHeight(), Image< T >::getWidth(), Image< T >::initialized(), and Image< T >::setVal().

**10.122.4.105** `template<class T> void Image< T >::splitPosNeg (Image< T > & pos, Image< T > & neg)`

Put all values >= 0 into pos and the negated of all vals <= 0 into neg.

Definition at line 261 of file Image\_MonoOnly.C.

References ASSERT, Image< T >::begin(), Image< T >::beginw(), Image< T >::const\_iterator, Image< T >::end(), Image< T >::getDims(), Image< T >::initialized(), Image< T >::iterator, NO\_INIT, and Image< T >::resize().

**10.122.4.106** `template<class T> void Image< T >::square ()`

Take square of all elements.

Definition at line 190 of file Image.C.

References ASSERT, Image< T >::beginw(), Image< T >::endw(), Image< T >::initialized(), and Image< T >::iterator.

**10.122.4.107** `template<class T> void Image< T >::swap (Image< T > & other) [inline]`

Swap the contents of two images.

Definition at line 718 of file Image.H.

References Image< T >::itsHdl.

Referenced by Image< T >::freeMem(), Jet< T >::operator=(), and Image< T >::operator=().

**10.122.4.108** `template<class T> void Image< T >::varianceNormalize (const T mi, const T ma)`

Normalize between *mi* and *ma* and multiply by variance.

Definition at line 221 of file `Image_MonoOnly.C`.

References `ASSERT`, `Image< T >::getStdev()`, `Image< T >::initialized()`, and `Image< T >::normalize()`.

**10.122.4.109** `template<class T> void Image< T >::wilsonNormalize (const float H, const float eps)`

Wilson non-linearity.

Definition at line 315 of file `Image_MonoOnly.C`.

References `ASSERT`, `Image< T >::beginw()`, `Image< T >::endw()`, `Image< T >::initialized()`, and `Image< T >::iterator`.

**10.122.4.110** `template<class T> void Image< T >::writeText (const Point2D & top_left, const char * text, const T col = T(), const T bgcol = T(255))`

Write some text; only ASCII chars between 32 and 126 are supported.

Definition at line 584 of file `Image.C`.

References `Image< T >::getHeight()`, `Image< T >::getWidth()`, `Point2D::i`, `Point2D::j`, and `Image< T >::setVal()`.

Referenced by `PsychoDisplay::displayText()`, `SimulationViewerStd::drawMegaCombo()`, `VisualEventSet::drawTokens()`, `main()`, and `normalizeWithScale()`.

The documentation for this class was generated from the following files:

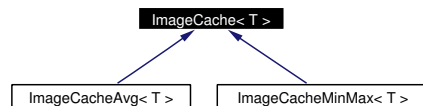
- [Image.H](#)
- [Image.C](#)
- [Image\\_MonoOnly.C](#)

## 10.123 ImageCache< T > Class Template Reference

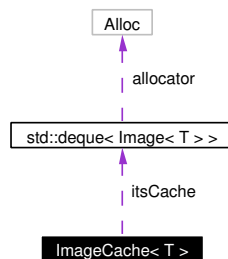
base class for image caches that do computations on the fly

```
#include <ImageCache.H>
```

Inheritance diagram for ImageCache< T >:



Collaboration diagram for ImageCache< T >:



### Public Member Functions

- **ImageCache** (uint maxSize=0)  
*Constructor.*
- virtual **~ImageCache** ()  
*Destructor.*
- virtual void **setMaxSize** (const uint maxSize)  
*Set maximum number of images in the cache.*
- virtual uint **getMaxSize** () const  
*Get maximum number of images in the cache.*
- virtual void **push\_back** (const Image< T > &img)  
*add image to cache - if the cache gets to big, old images are popped off*

- virtual `Image< T > pop_front ()`  
*pop the front `Image` (oldest) off the cache and return it*
- virtual const `Image< T > & back () const`  
*access the last `Image` (newest) in the queue*
- virtual const `Image< T > & front () const`  
*access the first `Image` (oldest) in the queue*
- virtual const `Image< T > & getImage (const uint lev) const`  
*Get image from a given level.*
- virtual const `Image< T > & operator[] (const uint lev) const`  
*Get image from a given level (shorthand for `getImage()`).*
- virtual `uint size () const`  
*return the current size of the cache*
- virtual bool `empty () const`  
*true if the cache contains no elements*
- virtual void `clear ()`  
*clear the cache, preserving its `maxSize`*

### Protected Member Functions

- virtual void `popOffOld ()`  
*checks whether the cache has become too big and removes old entries*
- virtual void `doWhenAdd (const Image< T > &img)`  
*called when an image is added - override in your derived classes!*
- virtual void `doWhenRemove (const Image< T > &img)`  
*called when an image is removed - override in your derived classes*

### Protected Attributes

- `uint itsMaxSize`  
*the maximum size of images to be stored*



- `std::deque< Image< T > > itsCache`  
the cache of the images

### 10.123.1 Detailed Description

**template<class T> class ImageCache< T >**

base class for image caches that do computations on the fly

This base class has no op doWhenAdd and doWhenRemove functions that should be overridden in classes derived from this one.

Definition at line 52 of file ImageCache.H.

### 10.123.2 Constructor & Destructor Documentation

**10.123.2.1** **template<class T> ImageCache< T >::ImageCache (uint maxSize = 0)**

Constructor.

**Parameters:**

**maxSize** the maximum size of the cache. If this size is exceeded, images are popped off the front of the cache and disregarded for the computation of the mean. If maxSize = 0, the cache is not limited.

Definition at line 46 of file ImageCache.C.

References uint.

**10.123.2.2** **template<class T> ImageCache< T >::~ImageCache ()**  
[virtual]

Destructor.

Definition at line 52 of file ImageCache.C.

### 10.123.3 Member Function Documentation

**10.123.3.1** **template<class T> const Image< T > & ImageCache< T >::back ()**  
const [virtual]

access the last Image (newest) in the queue

Definition at line 105 of file ImageCache.C.

References ASSERT.

#### 10.123.3.2 `template<class T> void ImageCache< T >::clear () [virtual]`

clear the cache, preserving its maxSize

Definition at line 144 of file ImageCache.C.

References ImageCache< T >::pop\_front(), and ImageCache< T >::size().

#### 10.123.3.3 `template<class T> void ImageCache< T >::doWhenAdd (const Image< T > & img) [protected, virtual]`

called when an image is added - override in your derived classes!

in ImageCache, this function is no op

Reimplemented in [ImageCacheAvg< T >](#), [ImageCacheMinMax< T >](#), [ImageCacheAvg< float >](#), [ImageCacheAvg< byte >](#), and [ImageCacheMinMax< float >](#).

Definition at line 154 of file ImageCache.C.

Referenced by ImageCache< T >::push\_back().

#### 10.123.3.4 `template<class T> void ImageCache< T >::doWhenRemove (const Image< T > & img) [protected, virtual]`

called when an image is removed - override in your derived classes

in ImageCache, this function is no op

Reimplemented in [ImageCacheAvg< T >](#), [ImageCacheMinMax< T >](#), [ImageCacheAvg< float >](#), [ImageCacheAvg< byte >](#), and [ImageCacheMinMax< float >](#).

Definition at line 159 of file ImageCache.C.

Referenced by ImageCache< T >::pop\_front().

#### 10.123.3.5 `template<class T> bool ImageCache< T >::empty () const [virtual]`

true if the cache contains no elements

Definition at line 139 of file ImageCache.C.

**10.123.3.6** `template<class T> const Image< T > & ImageCache< T >::front() const [virtual]`

access the first [Image](#) (oldest) in the queue

Definition at line 113 of file ImageCache.C.

References ASSERT.

**10.123.3.7** `template<class T> const Image< T > & ImageCache< T >::getImage(const uint lev) const [virtual]`

Get image from a given level.

Definition at line 121 of file ImageCache.C.

References ASSERT, and uint.

Referenced by ImageCache< T >::operator[ ]().

**10.123.3.8** `template<class T> uint ImageCache< T >::getMaxSize() const [virtual]`

Get maximum number of images in the cache.

Definition at line 66 of file ImageCache.C.

References ImageCache< T >::itsMaxSize, and uint.

**10.123.3.9** `]`

`template<class T> const Image< T > & ImageCache< T >::operator[ ](const uint lev) const [virtual]`

Get image from a given level (shorthand for [getImage\(\)](#)).

Definition at line 129 of file ImageCache.C.

References ImageCache< T >::getImage(), and uint.

**10.123.3.10** `template<class T> Image< T > ImageCache< T >::pop_front() [virtual]`

pop the front [Image](#) (oldest) off the cache and return it

Definition at line 94 of file ImageCache.C.

References ASSERT, and ImageCache< T >::doWhenRemove().

Referenced by ImageCache< T >::clear(), and ImageCache< T >::popOffOld().

#### 10.123.3.11 **template<class T> void [ImageCache](#)< T >::popOffOld ()** [protected, virtual]

checks whether the cache has become too big and removes old entries

Definition at line 81 of file ImageCache.C.

References [ImageCache](#)< T >::itsMaxSize, and [ImageCache](#)< T >::pop\_front().

Referenced by [ImageCache](#)< T >::push\_back(), and [ImageCache](#)< T >::setMaxSize().

#### 10.123.3.12 **template<class T> void [ImageCache](#)< T >::push\_back (const [Image](#)< T > &img)** [virtual]

add image to cache - if the cache gets to big, old images are popped off

Definition at line 71 of file ImageCache.C.

References [ImageCache](#)< T >::doWhenAdd(), and [ImageCache](#)< T >::popOffOld().

#### 10.123.3.13 **template<class T> void [ImageCache](#)< T >::setMaxSize (const [uint](#) maxSize)** [virtual]

Set maximum number of images in the cache.

If the cache currently has more images than the new specified maximum, it will be truncated to that maximum.

Definition at line 57 of file ImageCache.C.

References [ImageCache](#)< T >::itsMaxSize, [ImageCache](#)< T >::popOffOld(), and [uint](#).

#### 10.123.3.14 **template<class T> [uint](#) [ImageCache](#)< T >::size () const** [virtual]

return the current size of the cache

This may be smaller than the maximum size specified at construction or with [setMaxSize\(\)](#), if the cache is not full yet.

Definition at line 134 of file ImageCache.C.

References [uint](#).

Referenced by [ImageCache](#)< T >::clear().

## 10.123.4 Member Data Documentation

### 10.123.4.1 `template<class T> std::deque< Image<T> > ImageCache< T >::itsCache [protected]`

the cache of the images

Definition at line 117 of file ImageCache.H.

### 10.123.4.2 `template<class T> uint ImageCache< T >::itsMaxSize [protected]`

the maximum size of images to be stored

Definition at line 114 of file ImageCache.H.

Referenced by `ImageCache< T >::getMaxSize()`, `ImageCache< T >::popOffOld()`, and `ImageCache< T >::setMaxSize()`.

The documentation for this class was generated from the following files:

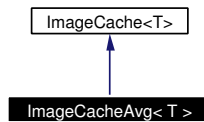
- [ImageCache.H](#)
- [ImageCache.C](#)

## 10.124 ImageCacheAvg< T > Class Template Reference

image cache to compute the running average

```
#include <ImageCache.H>
```

Inheritance diagram for ImageCacheAvg< T >:



Collaboration diagram for ImageCacheAvg< T >:

### Public Member Functions

- [ImageCacheAvg \(\)](#)  
*Uninitialized constructor.*
- [ImageCacheAvg \(uint maxSize\)](#)  
*Constructor.*
- [Image< T > mean \(\) const](#)  
*return the mean of the images currently in the cache*
- [Image< T > absDiffMean \(const Image< T > &img\) const](#)  
*return abs(img - mean)*
- [Image< T > clampedDiffMean \(const Image< T > &img\) const](#)  
*return (img - mean), clamped to zero where negative*
- [Image< typename promote\\_trait< T, float >::TP > sum \(\) const](#)  
*returns the sum of all cached images*

### Protected Member Functions

- virtual void [doWhenAdd](#) (const [Image< T >](#) &img)  
*called when an image is added - add image to the sum*

- virtual void [doWhenRemove](#) (const [Image](#)< T > &img)  
*called when an image is removed - subtract image from the sum*

## Protected Attributes

- [Image](#)< typename [promote\\_trait](#)< T, float >::TP > [itsSumImg](#)  
*the sum of all the cached images*

### 10.124.1 Detailed Description

**template<class T> class ImageCacheAvg< T >**

image cache to compute the running average

Definition at line 123 of file ImageCache.H.

### 10.124.2 Constructor & Destructor Documentation

**10.124.2.1 template<class T> [ImageCacheAvg](#)< T >::ImageCacheAvg ()**

Uninitialized constructor.

By default, cache size is not limited. A limit can be set later using [setMaxSize\(\)](#).

Definition at line 166 of file ImageCache.C.

**10.124.2.2 template<class T> [ImageCacheAvg](#)< T >::ImageCacheAvg (uint *maxSize*)**

Constructor.

#### Parameters:

***maxSize*** the maximum size of the cache. If this size is exceeded, images are popped off the front of the cache and disregarded for the computation of the mean. If *maxSize* = 0, the cache is not limited.

Definition at line 172 of file ImageCache.C.

References [uint](#).

### 10.124.3 Member Function Documentation

**10.124.3.1** `template<class T> Image< T > ImageCacheAvg< T >::absDiffMean (const Image< T > & img) const`

return abs(img - mean)

Definition at line 185 of file ImageCache.C.

References [absDiff\(\)](#), and [ImageCacheAvg](#)< T >::mean().

**10.124.3.2** `template<class T> Image< T > ImageCacheAvg< T >::clampedDiffMean (const Image< T > & img) const`

return (img - mean), clamped to zero where negative

Definition at line 192 of file ImageCache.C.

References [clampedDiff\(\)](#), and [ImageCacheAvg](#)< T >::mean().

**10.124.3.3** `template<class T> void ImageCacheAvg< T >::doWhenAdd (const Image< T > & img) [protected, virtual]`

called when an image is added - add image to the sum

Reimplemented from [ImageCache](#)< T >.

Definition at line 206 of file ImageCache.C.

References [ASSERT](#), and [ImageCacheAvg](#)< T >::itsSumImg.

**10.124.3.4** `template<class T> void ImageCacheAvg< T >::doWhenRemove (const Image< T > & img) [protected, virtual]`

called when an image is removed - subtract image from the sum

Reimplemented from [ImageCache](#)< T >.

Definition at line 221 of file ImageCache.C.

References [ASSERT](#), and [ImageCacheAvg](#)< T >::itsSumImg.

**10.124.3.5** `template<class T> Image< T > ImageCacheAvg< T >::mean () const`

return the mean of the images currently in the cache

Definition at line 178 of file ImageCache.C.



References ImageCacheAvg< T >::itsSumImg.

Referenced by ImageCacheAvg< T >::absDiffMean(), and ImageCacheAvg< T >::clampedDiffMean().

#### 10.124.3.6 `template<class T> Image< typename promote_trait< T, float >::TP > ImageCacheAvg< T >::sum () const`

returns the sum of all cached images

This will be of type `Image<int>` if `T = byte, int16 or int32`; of type `float` if `T = float`; of type `Image< PixRGB<int> >` if `T = PixRGB<byte>, PixRGB<int16> or PixRGB<int32>`; and of type `Image< PixRGB<float> >` if `T = PixRGB<float>`

Definition at line 199 of file ImageCache.C.

References ImageCacheAvg< T >::itsSumImg.

### 10.124.4 Member Data Documentation

#### 10.124.4.1 `template<class T> Image<typename promote_trait<T,float>::TP> ImageCacheAvg< T >::itsSumImg` [protected]

the sum of all the cached images

Definition at line 161 of file ImageCache.H.

Referenced by ImageCacheAvg< T >::doWhenAdd(), ImageCacheAvg< T >::doWhenRemove(), ImageCacheAvg< T >::mean(), and ImageCacheAvg< T >::sum().

The documentation for this class was generated from the following files:

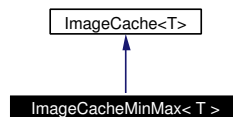
- [ImageCache.H](#)
- [ImageCache.C](#)

## 10.125 ImageCacheMinMax< T > Class Template Reference

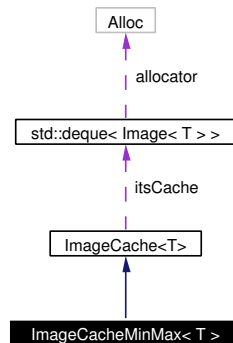
image cache to compute a running min/max

```
#include <ImageCache.H>
```

Inheritance diagram for ImageCacheMinMax< T >:



Collaboration diagram for ImageCacheMinMax< T >:



### Public Member Functions

- [ImageCacheMinMax \(\)](#)  
*Uninitialized constructor.*
- [ImageCacheMinMax \(uint maxSize\)](#)  
*Constructor.*
- [Image< T > getMax \(\) const](#)  
*return the max of the images currently in the cache*
- [Image< T > getMin \(\) const](#)  
*return the max of the images currently in the cache*

## Protected Member Functions

- virtual void `doWhenAdd` (const `Image< T > &img`)  
*called when an image is added - here it's a nop-op*
- virtual void `doWhenRemove` (const `Image< T > &img`)  
*called when an image is removed - here it's a no-op*

### 10.125.1 Detailed Description

**template<class T> class ImageCacheMinMax< T >**

image cache to compute a running min/max

In this implementation, all the computations are done when `getMax()` or `getMin()` are called. That is, the cache is fairly dump and does not attempt to keep an updated min and max image as new images get added or removed. This differs substantially from the `ImageCacheAvg` implementation, where all computations are done as images are added/removed and getting the mean is a very cheap operation.

Definition at line 173 of file `ImageCache.H`.

### 10.125.2 Constructor & Destructor Documentation

**10.125.2.1** **template<class T> ImageCacheMinMax< T >::ImageCacheMinMax ()**

Uninitialized constructor.

By default, cache size is not limited. A limit can be set later using `setMaxSize()`.

Definition at line 232 of file `ImageCache.C`.

**10.125.2.2** **template<class T> ImageCacheMinMax< T >::ImageCacheMinMax (uint maxSize)**

Constructor.

#### Parameters:

**maxSize** the maximum size of the cache. If this size is exceeded, images are popped off the front of the cache and disregarded for the computation of the mean. If `maxSize = 0`, the cache is not limited.

Definition at line 238 of file `ImageCache.C`.

References uint.

### 10.125.3 Member Function Documentation

**10.125.3.1** `template<class T> void ImageCacheMinMax< T >::doWhenAdd  
(const Image< T > & img) [protected, virtual]`

called when an image is added - here it's a nop-op

Reimplemented from [ImageCache< T >](#).

Definition at line 268 of file ImageCache.C.

**10.125.3.2** `template<class T> void ImageCacheMinMax< T  
>::doWhenRemove (const Image< T > & img) [protected,  
virtual]`

called when an image is removed - here it's a no-op

Reimplemented from [ImageCache< T >](#).

Definition at line 273 of file ImageCache.C.

**10.125.3.3** `template<class T> Image< T > ImageCacheMinMax< T  
>::getMax () const`

return the max of the images currently in the cache

Definition at line 244 of file ImageCache.C.

References takeMax().

**10.125.3.4** `template<class T> Image< T > ImageCacheMinMax< T  
>::getMin () const`

return the max of the images currently in the cache

Definition at line 256 of file ImageCache.C.

References takeMin().

The documentation for this class was generated from the following files:

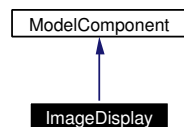
- [ImageCache.H](#)
- [ImageCache.C](#)

## 10.126 ImageDisplay Class Reference

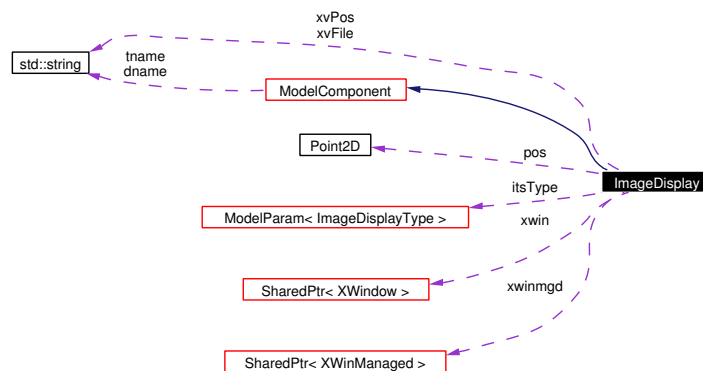
A generalized class for displaying images.

```
#include <ImageDisplay.H>
```

Inheritance diagram for ImageDisplay:



Collaboration diagram for ImageDisplay:



### Public Member Functions

- **ImageDisplay** (**ModelManager** &mgr, const **std::string** &descrName="Image Display", const **std::string** &tagName="ImageDisplay", int x=-1, int y=-1)  
*Constructor.*
- **~ImageDisplay** ()  
*Destructor.*
- virtual void **exportOptions** (const int whatToExport, const bool recurse)  
*Export some of our **ModelParam** parameters as command-line options.*
- **ImageDisplayType** **getType** (void)

*Get what display type we are using.*

- void [display](#) (const [Image](#)< [PixRGB](#)< [byte](#) > > &img)

*Display an RGB image.*

- void [display](#) (const [Image](#)< [byte](#) > &img)

*Display a grayscale image.*

- void [close](#) (void)

*Close display.*

- void [setPosition](#) (int x, int y)

*Set position of display window.*

- [Point2D](#) [getPosition](#) (void)

*Get position of display window.*

- void [enableXVUpdate](#) (void)

*Allow XV to receive SIGQUIT to update the image (unreliable sometimes).*

- void [disableXVUpdate](#) (void)

*Do not allow XV to receive SIGQUIT to update the image (default).*

## Protected Attributes

- [ModelParam](#)< [ImageDisplayType](#) > itsType

*type of image display to use*

### 10.126.1 Detailed Description

A generalized class for displaying images.

The ImageDisplay class provides a generic [display\(\)](#) function that can display images in a manner specified from the command-line as None, XV, [XWindow](#), or [XWinManaged](#).

Definition at line 47 of file ImageDisplay.H.

## 10.126.2 Constructor & Destructor Documentation

### 10.126.2.1 ImageDisplay::ImageDisplay ([ModelManager](#) & *mgr*, const [std::string](#) & *descrName* = "Image Display", const [std::string](#) & *tagName* = "ImageDisplay", int *x* = -1, int *y* = -1)

Constructor.

Unlike the constructors for [XWindow](#) and [XWinManaged](#), the constructor for ImageDisplay will not create the display window first and wait for an image to be drawn. Rather, the [display\(\)](#) function will check if the display is already open when the image needs to be drawn. This way the user can specify the type of display needed.

#### Parameters:

- mgr* our model manager; see [ModelManager.H](#)
- descrName* a human-readable descriptive name
- tagname* a computer-oriented unique name (used for storage in [ParamMap](#))
- x* horizontal position hint for window placement (use -1 for automatic placement)
- y* vertical position hint for window placement (use -1 for automatic placement)

Definition at line 39 of file ImageDisplay.C.

References [DISPLAY\\_NONE](#), and [ImageDisplayType](#).

### 10.126.2.2 ImageDisplay::~ImageDisplay ()

Destructor.

Definition at line 60 of file ImageDisplay.C.

## 10.126.3 Member Function Documentation

### 10.126.3.1 void ImageDisplay::close (void)

Close display.

Definition at line 191 of file ImageDisplay.C.

References [DISPLAY\\_NONE](#), [DISPLAY\\_XV](#), [DISPLAY\\_XWIN](#), [DISPLAY\\_XWINMGD](#), [ModelParam< ImageDisplayType >::getVal\(\)](#), [itsType](#), [SharedPtr< XWinManaged >::reset\(\)](#), and [SharedPtr< XWindow >::reset\(\)](#).

### 10.126.3.2 void ImageDisplay::disableXVUpdate (void)

Do not allow XV to receive SIGQUIT to update the image (default).

Definition at line 264 of file ImageDisplay.C.

### 10.126.3.3 void ImageDisplay::display (const Image< byte > & img)

Display a grayscale image.

Definition at line 139 of file ImageDisplay.C.

References ModelComponent::descriptiveName(), DISPLAY\_NONE, DISPLAY\_XV, DISPLAY\_XWIN, DISPLAY\_XWINMGD, Image< T >::getHeight(), ModelParam< ImageDisplayType >::getVal(), Image< T >::getWidth(), Point2D::i, itsType, Point2D::j, SharedPtr< XWinManaged >::reset(), SharedPtr< XWindow >::reset(), ModelComponent::tagName(), and Raster::WriteGray().

### 10.126.3.4 void ImageDisplay::display (const Image< PixRGB< byte > > & img)

Display an RGB image.

Definition at line 87 of file ImageDisplay.C.

References ModelComponent::descriptiveName(), DISPLAY\_NONE, DISPLAY\_XV, DISPLAY\_XWIN, DISPLAY\_XWINMGD, ModelParam< ImageDisplayType >::getVal(), Point2D::i, itsType, Point2D::j, SharedPtr< XWinManaged >::reset(), SharedPtr< XWindow >::reset(), ModelComponent::tagName(), and Raster::WriteRGB().

### 10.126.3.5 void ImageDisplay::enableXVUpdate (void)

Allow XV to receive SIGQUIT to update the image (unreliable sometimes).

Definition at line 259 of file ImageDisplay.C.

### 10.126.3.6 void ImageDisplay::exportOptions (const int *whatToExport*, const bool *recurse*) [virtual]

Export some of our [ModelParam](#) parameters as command-line options.

See the base function in [ModelComponent.H](#) for info.

Reimplemented from [ModelComponent](#).

Definition at line 66 of file ImageDisplay.C.

References ModelComponent::exportOptions(), itsType, OPTEXP\_CORE, and ModelManager::requestOption().



### 10.126.3.7 [Point2D](#) ImageDisplay::getPosition (void)

Get position of display window.

Definition at line 252 of file ImageDisplay.C.

### 10.126.3.8 [ImageDisplayType](#) ImageDisplay::getType (void)

Get what display type we are using.

Definition at line 80 of file ImageDisplay.C.

References [ModelParam< ImageDisplayType >::getVal\(\)](#), [ImageDisplayType](#), and [itsType](#).

### 10.126.3.9 void ImageDisplay::setPosition (int x, int y)

Set position of display window.

Attempts to set the upper-left coordinate of display window.

#### Parameters:

- x horizontal position hint for window placement (use 0 for automatic placement)
- y vertical position hint for window placement (use 0 for automatic placement)

Definition at line 217 of file ImageDisplay.C.

References [DISPLAY\\_NONE](#), [DISPLAY\\_XV](#), [DISPLAY\\_XWIN](#), [DISPLAY\\_XWINMGD](#), [ModelParam< ImageDisplayType >::getVal\(\)](#), [Point2D::i](#), [itsType](#), and [Point2D::j](#).

## 10.126.4 Member Data Documentation

### 10.126.4.1 [ModelParam<ImageDisplayType>](#) ImageDisplay::itsType [protected]

type of image display to use

Definition at line 103 of file ImageDisplay.H.

Referenced by [close\(\)](#), [display\(\)](#), [exportOptions\(\)](#), [getType\(\)](#), and [setPosition\(\)](#).

The documentation for this class was generated from the following files:

- [ImageDisplay.H](#)
- [ImageDisplay.C](#)

## 10.127 ImageMap Class Reference

3D image map created from the image at diff' orientations

```
#include <ImageMap.H>
```

### Public Attributes

- float [pix](#) [AnglesUsed][ImageSizeX][ImageSizeY]  
*value of pixel*
- bool [zero](#) [AnglesUsed][ImageSizeX][ImageSizeY]  
*is true if this is a non-zero value*

### 10.127.1 Detailed Description

3D image map created from the image at diff' orientations

Definition at line 50 of file ImageMap.H.

### 10.127.2 Member Data Documentation

#### 10.127.2.1 float [ImageMap::pix](#)[AnglesUsed][ImageSizeX][ImageSizeY]

value of pixel

Definition at line 54 of file ImageMap.H.

#### 10.127.2.2 bool [ImageMap::zero](#)[AnglesUsed][ImageSizeX][ImageSizeY]

is true if this is a non-zero value

Definition at line 56 of file ImageMap.H.

The documentation for this class was generated from the following file:

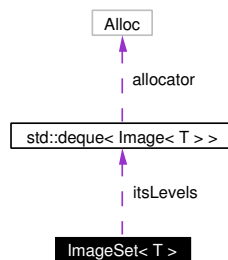
- [ImageMap.H](#)

## 10.128 ImageSet< T > Class Template Reference

This class implements a set of images, often used as a dyadic pyramid.

```
#include <ImageSet.H>
```

Collaboration diagram for ImageSet< T >:



### Public Member Functions

- `ImageSet` (const `uint` n=0)  
Construct with a given number of empty images.
- `ImageSet` (const `uint` n, const `Dims` &dims, `InitPolicy` init=ZEROS)  
Construct with a given number of images of the given size.
- void `swap` (`ImageSet`< T > &that)  
Swap contents with another `ImageSet`.
- void `clear` ()  
Keep the same number of images, but make all images empty (0-by-0 size).
- void `reset` (`uint` newDepth=0)  
Reset to a new number of images, all images will be empty (0-by-0 size).
- `uint` `size` () const  
Return number of images in image set.
- `uint` `getDepth` () const  
A synonym for `size()` (to be backward-compatible with Pyramid syntax).
- const `Image`< T > & `getImage` (const `uint` lev) const  
Get image from a given level.

- `const Image< T > & operator[ ] (const uint lev) const`  
*Get image from a given level (shorthand for `getImage()`).*
- `Image< T > & getImageMut (const uint lev)`  
*Get mutable image from a given level.*
- `Image< T > & operator[ ] (const uint lev)`  
*Get mutable image from a given level (shorthand for `getImageMut()`).*
- `T getPixel (const Point2D &p, const uint lev) const`  
*Get value of a pixel at a given level and coordinates.*
- `T getPixelNI (const Point2D &p, const uint lev) const`  
*Get value of a pixel at a given level and coordinates, no interpolation.*
- `void push_back (Image< T > &img)`  
*push an image to the back of the set*
- `void push_front (Image< T > &img)`  
*push an image to the front of the set*
- `Image< T > pop_back ()`  
*return the last image in the set and pop it off*
- `Image< T > pop_front ()`  
*return the first image in the set and pop it off*
- `Image< T > front () const`  
*return the last image in the set (without popping it off)*
- `Image< T > back () const`  
*return the first image in the set (without popping it off)*
- `bool isEmpty () const`  
*Return true if the pyramid has no non-empty images ( $width*height > 0$ ).*
- `bool isNonEmpty () const`  
*Return true if the pyramid has any non-empty images ( $width*height > 0$ ).*

### 10.128.1 Detailed Description

**template<class T> class ImageSet< T >**

This class implements a set of images, often used as a dyadic pyramid.

Definition at line 51 of file ImageSet.H.

### 10.128.2 Constructor & Destructor Documentation

**10.128.2.1    template<class T> ImageSet< T >::ImageSet (const uint n = 0)**

Construct with a given number of empty images.

Definition at line 48 of file ImageSet.C.

References uint.

**10.128.2.2    template<class T> ImageSet< T >::ImageSet (const uint n, const Dims & dims, InitPolicy init = ZEROS)**

Construct with a given number of images of the given size.

Definition at line 54 of file ImageSet.C.

References uint.

### 10.128.3 Member Function Documentation

**10.128.3.1    template<class T> Image< T > ImageSet< T >::back () const**

return the first image in the set (without popping it off)

Definition at line 162 of file ImageSet.C.

References ASSERT, and ImageSet< T >::isNonEmpty().

**10.128.3.2    template<class T> void ImageSet< T >::clear ()**

Keep the same number of images, but make all images empty (0-by-0 size).

This returns the ImageSet to a state where isEmpty() is true.

Definition at line 68 of file ImageSet.C.

Referenced by doClear().

**10.128.3.3    template<class T> [Image](#)< T > [ImageSet](#)< T >::front () const**

return the last image in the set (without popping it off)

Definition at line 170 of file [ImageSet.C](#).

References [ASSERT](#), and [ImageSet](#)< T >::isEmpty().

**10.128.3.4    template<class T> [uint](#) [ImageSet](#)< T >::getDepth () const  
[inline]**

A synonym for [size\(\)](#) (to be backward-compatible with Pyramid syntax).

Definition at line 144 of file [ImageSet.H](#).

References [uint](#).

Referenced by [buildPyrGabor\(\)](#), [centerSurround\(\)](#), [centerSurroundDiff\(\)](#), [centerSurroundDiffSingleOpponent\(\)](#), [centerSurroundSingleOpponent\(\)](#), [doEnergyNorm\(\)](#), [isDyadic\(\)](#), and [ImageSet](#)< T >::isEmpty().

**10.128.3.5    template<class T> const [Image](#)< T > & [ImageSet](#)< T >::getImage  
(const [uint](#) lev) const**

Get image from a given level.

Definition at line 83 of file [ImageSet.C](#).

References [ASSERT](#), and [uint](#).

Referenced by [ImageSet](#)< T >::operator[]().

**10.128.3.6    template<class T> [Image](#)< T > & [ImageSet](#)< T >::getImageMut  
(const [uint](#) lev)**

Get mutable image from a given level.

Definition at line 91 of file [ImageSet.C](#).

References [ASSERT](#), and [uint](#).

Referenced by [ImageSet](#)< T >::operator[]().

**10.128.3.7    template<class T> T [ImageSet](#)< T >::getPixel (const [Point2D](#) & p,  
const [uint](#) lev) const**

Get value of a pixel at a given level and coordinates.

**Parameters:**

- pix* The pixel value returned
- p* Coordinates given at scale 0 (base pyramid level); bilinear interpolation will be used to compute pixel value at given level.
- lev* The pyramid level to get the pixel value from.

Definition at line 99 of file ImageSet.C.

References ASSERT, and uint.

Referenced by main(), and BeobotVisualCortex::singleCPUprocess().

### 10.128.3.8 `template<class T> T ImageSet< T >::getPixelNI (const Point2D & p, const uint lev) const`

Get value of a pixel at a given level and coordinates, no interpolation.

**Parameters:**

- pix* The pixel value returned
- p* Coordinates given at scale 0 (base pyramid level); This version does not use interpolation.
- lev* The pyramid level to get the pixel value from.

Definition at line 114 of file ImageSet.C.

References ASSERT, and uint.

### 10.128.3.9 `template<class T> bool ImageSet< T >::isEmpty () const [inline]`

Return true if the pyramid has no non-empty images (width\*height > 0).

Definition at line 161 of file ImageSet.H.

References ImageSet< T >::isNonEmpty().

Referenced by ImgVec2mexArr().

### 10.128.3.10 `template<class T> bool ImageSet< T >::isNonEmpty () const`

Return true if the pyramid has any non-empty images (width\*height > 0).

Definition at line 178 of file ImageSet.C.

References ImageSet< T >::getDepth(), and uint.

Referenced by `ImageSet< T >::back()`, `centerSurround()`, `centerSurroundDiff()`, `centerSurroundDiffSingleOpponent()`, `centerSurroundSingleOpponent()`, `ImageSet< T >::front()`, `ImageSet< T >::isEmpty()`, `ImageSet< T >::pop_back()`, and `ImageSet< T >::pop_front()`.

#### 10.128.3.11 `ImageMut()`

```
template<class T> Image< T > & ImageSet< T >::operator[] (const uint lev)
[inline]
```

Get mutable image from a given level (shorthand for `getImageMut()`).

Definition at line 157 of file `ImageSet.H`.

References `ImageSet< T >::getImageMut()`, and `uint`.

#### 10.128.3.12 `Image()`

```
template<class T> const Image< T > & ImageSet< T >::operator[] (const uint lev)
const [inline]
```

Get image from a given level (shorthand for `getImage()`).

Definition at line 152 of file `ImageSet.H`.

References `ImageSet< T >::getImage()`, and `uint`.

#### 10.128.3.13 `template<class T> Image< T > ImageSet< T >::pop_back ()`

return the last image in the set and pop it off

Definition at line 142 of file `ImageSet.C`.

References `ASSERT`, and `ImageSet< T >::isNonEmpty()`.

#### 10.128.3.14 `template<class T> Image< T > ImageSet< T >::pop_front ()`

return the first image in the set and pop it off

Definition at line 152 of file `ImageSet.C`.

References `ASSERT`, and `ImageSet< T >::isNonEmpty()`.

#### 10.128.3.15 `template<class T> void ImageSet< T >::push_back (Image< T > & img)`

push an image to the back of the set



Definition at line 128 of file ImageSet.C.

**10.128.3.16** `template<class T> void ImageSet< T >::push_front (Image< T > & img)`

push an image to the front of the set

Definition at line 135 of file ImageSet.C.

**10.128.3.17** `template<class T> void ImageSet< T >::reset (uint newDepth = 0)`

Reset to a new number of images, all images will be empty (0-by-0 size).

Definition at line 76 of file ImageSet.C.

References uint.

**10.128.3.18** `template<class T> uint ImageSet< T >::size () const [inline]`

Return number of images in image set.

Definition at line 148 of file ImageSet.H.

References uint.

Referenced by TCPmessage::addImageSet(), clampedDiff(), doAddWeighted(), doApplyBiases(), doClear(), JunctionChannel::doInput(), doLowThresh(), doLowThreshAbs(), doOneNormalize(), doRectify(), doSqrt(), ImageSet2cellArr(), ImgVec2mexArr(), DirectFeedChannel::inputPyramid(), isHomogeneous(), makeImageArray(), mean(), operator \*(), operator +(), operator -(), operator /(), rangeOf(), reduce(), rescale(), SingleChannel::saveResults(), and sum().

**10.128.3.19** `template<class T> void ImageSet< T >::swap (ImageSet< T > & that)`

Swap contents with another ImageSet.

Definition at line 61 of file ImageSet.C.

References ImageSet< T >::itsLevels.

The documentation for this class was generated from the following files:

- [ImageSet.H](#)
- [ImageSet.C](#)

## 10.129 ImageSpring< T > Class Template Reference

This class is some sort of self organizing map used for clustering.

```
#include <ImageSpring.H>
```

Inheritance diagram for ImageSpring< T >:



Collaboration diagram for ImageSpring< T >:

### Public Member Functions

- [ImageSpring](#) (const int [width](#), const int [height](#), const int nbNeighbors)  
*Constructor that only allocates memory for given size.*
- [ImageSpring](#) ()  
*Constructor for uninitialized image (useful for arrays of Images).*
- [ImageSpring](#) (const [ImageSpring](#)< T > &A)  
*Copy constructor.*
- void [init](#) (const int [width](#), const int [height](#), const int nbNeighbors)  
*init function*
- void [init](#) (const T \*a, const float \*posX, const float \*posY, const float \*\*stiff, const int [width](#), const int [height](#), const int nbNeighbors)  
*init function, to copy another ImageSpring*
- [~ImageSpring](#) ()  
*Destructor.*
- void [freeMem](#) ()  
*Free memory and switch to uninitialized state.*
- void [initClustering](#) (bool initPosMasses)  
*initialize 'stiff' and 'pos'*

- void `computeStiff` (void)  
*compute the stiffness of the spring between the masses*
- void `computePos` (const float dt)  
*updates the positions of the masses*
- void `getStats` (void)  
*computes mean and stdev (\*\*heuristic\*\*)*
- void `getStatsDist` (void)  
*computes meanDist and stdevDist (\*\*heuristic\*\*)*
- void `getXY` (const int index, `Point2D` &pt) const  
*returns (X,Y) given an index=X+w\*Y (see posX)*
- void `getIndex` (const `Point2D` point, int &index) const  
*returns the index=X+w\*Y (see posX)*
- int `getNbNeighbors` (void) const  
*returns the number of neighbors*
- float `getDistanceMasses` (const int index1, const int index2) const  
*distance between the 2 masses*
- bool `getNeighbor` (const int index, const int n, int &indexNeighbor) const  
*computes the index of the nth neighbor of "index"*
- void `getPositions` (`Image`< `PixRGB`< byte > > &img, const int zoom=16)  
*Get position of masses and strength of springs (for debug).*
- void `getClusteredImage` (const `Image`< `PixRGB`< byte > > &scene, `Image`< `PixRGB`< byte > > &clusteredImage, `Point2D` &supposedTrackCentroid, const `Point2D` &previousTrackCentroid)  
*get clustered image and track centroid position*

### 10.129.1 Detailed Description

`template<class T> class ImageSpring< T >`

This class is some sort of self organizing map used for clustering.

All the pixels are linked by springs whose stiffness depends on the difference between them

Definition at line 49 of file ImageSpring.H.

## 10.129.2 Constructor & Destructor Documentation

### 10.129.2.1 `template<class T> ImageSpring< T >::ImageSpring (const int width, const int height, const int nbNeighbors) [inline]`

Constructor that only allocates memory for given size.

Definition at line 193 of file ImageSpring.H.

References `ImageSpring< T >::init()`, and `NO_INIT`.

### 10.129.2.2 `template<class T> ImageSpring< T >::ImageSpring () [inline]`

Constructor for uninitialized image (useful for arrays of Images).

Use `init()` later to initialize

Definition at line 202 of file ImageSpring.H.

### 10.129.2.3 `template<class T> ImageSpring< T >::ImageSpring (const ImageSpring< T > &A) [inline]`

Copy constructor.

Definition at line 207 of file ImageSpring.H.

References `ASSERT`, `Image< T >::getArrayPtr()`, `Image< T >::getHeight()`, `Image< T >::getWidth()`, `ImageSpring< T >::init()`, `Image< T >::initialized()`, `ImageSpring< T >::nbNeighbors`, and `ImageSpring< T >::stiff`.

### 10.129.2.4 `template<class T> ImageSpring< T >::~~ImageSpring () [inline]`

Destructor.

Definition at line 218 of file ImageSpring.H.

References `ImageSpring< T >::freeMem()`.

### 10.129.3 Member Function Documentation

#### 10.129.3.1 `template<class T> void ImageSpring< T >::computePos (const float dt)`

updates the positions of the masses

based on simple mechanics

Definition at line 163 of file ImageSpring.C.

References `Image< T >::getHeight()`, `ImageSpring< T >::getIndex()`, `ImageSpring< T >::getNeighbor()`, `Image< T >::getSize()`, `Image< T >::getWidth()`, and `Y`.

#### 10.129.3.2 `template<class T> void ImageSpring< T >::computeStiff (void)`

compute the stiffness of the spring between the masses

based on the color-distance between pixels

Definition at line 130 of file ImageSpring.C.

References `distance()`, `ImageSpring< T >::getNeighbor()`, `Image< T >::getSize()`, `Image< T >::getVal()`, and `ImageSpring< T >::getXY()`.

Referenced by `ImageSpring< T >::initClustering()`.

#### 10.129.3.3 `template<class T> void ImageSpring< T >::freeMem () [inline]`

Free memory and switch to uninitialized state.

Reimplemented from `Image< T >`.

Definition at line 223 of file ImageSpring.H.

References `Image< T >::freeMem()`, and `Image< T >::getSize()`.

Referenced by `ImageSpring< T >::init()`, and `ImageSpring< T >::~ImageSpring()`.

#### 10.129.3.4 `template<class T> void ImageSpring< T >::getClusteredImage (const Image< PixRGB< byte > > & scene, Image< PixRGB< byte > > & clusteredImage, Point2D & supposedTrackCentroid, const Point2D & previousTrackCentroid)`

get clustered image and track centroid position

Definition at line 279 of file ImageSpring.C.

References Image< T >::drawCross(), Image< T >::getHeight(), Image< T >::getVal(), Image< T >::getWidth(), Point2D::i, Point2D::j, NO\_INIT, PixRGB< T >::set(), Image< T >::setVal(), SQR(), sqrt(), and ZEROS.

#### 10.129.3.5 `template<class T> float ImageSpring< T >::getDistanceMasses (const int index1, const int index2) const`

distance between the 2 masses

Definition at line 211 of file ImageSpring.C.

References SQR(), and sqrt().

#### 10.129.3.6 `template<class T> void ImageSpring< T >::getIndex (const Point2D point, int & index) const [inline]`

returns the index=X+w\*Y (see posX)

Definition at line 289 of file ImageSpring.H.

References Image< T >::getWidth(), Point2D::i, and Point2D::j.

Referenced by ImageSpring< T >::computePos(), and ImageSpring< T >::getNeighbor().

#### 10.129.3.7 `template<class T> int ImageSpring< T >::getNbNeighbors (void) const [inline]`

returns the number of neighbors

Definition at line 347 of file ImageSpring.H.

#### 10.129.3.8 `template<class T> bool ImageSpring< T >::getNeighbor (const int index, const int n, int & indexNeighbor) const [inline]`

computes the index of the nth neighbor of "index"

returns true iff the neighbor is in the image

Definition at line 294 of file ImageSpring.H.

References ASSERT, Image< T >::coordsOk(), ImageSpring< T >::getIndex(), ImageSpring< T >::getXY(), Point2D::i, and Point2D::j.

Referenced by ImageSpring< T >::computePos(), ImageSpring< T >::computeStiff(), ImageSpring< T >::getPositions(), and ImageSpring< T >::getStatsDist().

### 10.129.3.9 `template<class T> void ImageSpring< T >::getPositions (Image< PixRGB< byte > > & img, const int zoom = 16)`

Get position of masses and strength of springs (for debug).

Definition at line 216 of file ImageSpring.C.

References Image< T >::getHeight(), ImageSpring< T >::getNeighbor(), and Image< T >::getSize().

### 10.129.3.10 `template<class T> void ImageSpring< T >::getStats (void)`

computes mean and stdev (\*\*heuristic\*\*)

this function is a heuristic because it will only consider nbHeuristic pixels at random

Definition at line 51 of file ImageSpring.C.

References ASSERT, Image< T >::getVal(), inverse(), randomUpTo(), and sqrt().

Referenced by ImageSpring< T >::initClustering().

### 10.129.3.11 `template<class T> void ImageSpring< T >::getStatsDist (void)`

computes meanDist and stdevDist (\*\*heuristic\*\*)

weight must be defined this function is an heuristic because it will only consider nb-Heuristic pairs of pixels at random

Definition at line 79 of file ImageSpring.C.

References ASSERT, distance(), ImageSpring< T >::getNeighbor(), Image< T >::getVal(), randomUpTo(), SQR(), and sqrt().

Referenced by ImageSpring< T >::initClustering().

### 10.129.3.12 `template<class T> void ImageSpring< T >::getXY (const int index, Point2D & pt) const [inline]`

returns (X,Y) given an index=X+w\*Y (see posX)

Definition at line 284 of file ImageSpring.H.

References Image< T >::getWidth(), Point2D::i, and Point2D::j.

Referenced by ImageSpring< T >::computeStiff(), and ImageSpring< T >::getNeighbor().

**10.129.3.13** `template<class T> void ImageSpring< T >::init (const T * a, const float * posX, const float * posY, const float ** stiff, const int width, const int height, const int nbNeighbors) [inline]`

init function, to copy another ImageSpring

Definition at line 238 of file ImageSpring.H.

References `ImageSpring< T >::freeMem()`, `Image< T >::getSize()`, and `Image< T >::resize()`.

**10.129.3.14** `template<class T> void ImageSpring< T >::init (const int width, const int height, const int nbNeighbors) [inline]`

init function

Definition at line 266 of file ImageSpring.H.

References `ImageSpring< T >::freeMem()`, `Image< T >::getSize()`, and `Image< T >::resize()`.

Referenced by `ImageSpring< T >::ImageSpring()`, and `BeobotVisualCortex::init()`.

**10.129.3.15** `template<class T> void ImageSpring< T >::initClustering (bool initPosMasses)`

initialize 'stiff' and 'pos'

Definition at line 112 of file ImageSpring.C.

References `ImageSpring< T >::computeStiff()`, `Image< T >::getHeight()`, `ImageSpring< T >::getStats()`, `ImageSpring< T >::getStatsDist()`, and `Image< T >::getWidth()`.

The documentation for this class was generated from the following files:

- [ImageSpring.H](#)
- [ImageSpring.C](#)

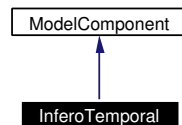


## 10.130 InferoTemporal Class Reference

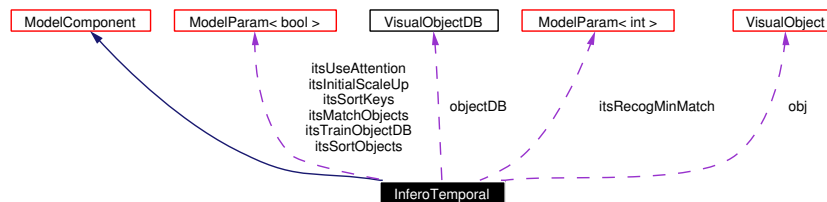
InferoTemporal.

```
#include <InferoTemporal.H>
```

Inheritance diagram for InferoTemporal:



Collaboration diagram for InferoTemporal:



### Public Member Functions

- **InferoTemporal** (**ModelManager** &mgr, const **std::string** &descrName, const **std::string** &tagName)  
*Unitialized constructor.*
- virtual **~InferoTemporal** ()  
*Destructor.*
- virtual void **exportOptions** (const int whatToExport, const bool recurse=true)  
*Export available options.*
- virtual void **freeMem** ()  
*Free all resources and get ready for a re- init().*
- virtual void **input** (**Image**< **PixRGB**< **byte** > > &objImg, **std::vector**< double > &feature, int attnShiftNum)  
*Object recognition.*

- virtual bool [getUseAttention](#) () const  
*Returns whether or not to mask the object image before input.*

## Protected Member Functions

- virtual void [start1](#) ()  
*(re-)initialization*
- bool [computeVisualObject](#) (const [Image](#)< float > &bwimg)  
*extract [VisualObject](#) from scene*

## Protected Attributes

- [ModelParam](#)< bool > **itsInitialScaleUp**
- [ModelParam](#)< bool > **itsUseAttention**
- [ModelParam](#)< [std::string](#) > **itsObjectDatabaseFile**
- [ModelParam](#)< bool > **itsTrainObjectDB**
- [ModelParam](#)< bool > **itsMatchObjects**
- [ModelParam](#)< int > **itsRecogMinMatch**
- [ModelParam](#)< [std::string](#) > **itsOutputFile**
- [ModelParam](#)< bool > **itsSortObjects**
- [ModelParam](#)< [std::string](#) > **itsMatchingAlg**
- [ModelParam](#)< bool > **itsSortKeys**
- [VisualObjectDB](#) **objectDB**
- [VisualObject](#) **obj**

### 10.130.1 Detailed Description

InferoTemporal.

Model of the Inferior Temporal Cortex utilizing SIFT object recognition

Definition at line 48 of file InferoTemporal.H.

### 10.130.2 Constructor & Destructor Documentation

#### 10.130.2.1 InferoTemporal::InferoTemporal ([ModelManager](#) & mgr, const [std::string](#) & descrName, const [std::string](#) & tagName)

Unitialized constructor.

Definition at line 51 of file InferoTemporal.C.

#### 10.130.2.2 InferoTemporal::~~InferoTemporal () [virtual]

Destructor.

Definition at line 77 of file InferoTemporal.C.

### 10.130.3 Member Function Documentation

#### 10.130.3.1 bool InferoTemporal::computeVisualObject (const Image< float > & *bwimg*) [protected]

extract VisualObject from scene

returns false if a VisualObject could not be extracted

Definition at line 172 of file InferoTemporal.C.

References [downSize\(\)](#), [ScaleSpace::findKeypoints\(\)](#), [Image< T >::getHeight\(\)](#), [ModelParam< bool >::getVal\(\)](#), [Image< T >::getWidth\(\)](#), [ScaleSpace::initScaleSpace\(\)](#), and [rescale\(\)](#).

Referenced by [input\(\)](#).

#### 10.130.3.2 void InferoTemporal::exportOptions (const int *whatToExport*, const bool *recurse* = true) [virtual]

Export available options.

Reimplemented from [ModelComponent](#).

Definition at line 82 of file InferoTemporal.C.

References [ModelComponent::exportOptions\(\)](#), [OPTEXP\\_CORE](#), and [ModelManager::requestOption\(\)](#).

#### 10.130.3.3 void InferoTemporal::freeMem () [virtual]

Free all resources and get ready for a re- init().

Definition at line 101 of file InferoTemporal.C.

#### 10.130.3.4 bool InferoTemporal::getUseAttention () const [virtual]

Returns whether or not to mask the object image before input.

Definition at line 166 of file InferenceTemporal.C.

References `ModelParam< bool >::getVal()`.

**10.130.3.5** `void InferenceTemporal::input (Image< PixRGB< byte > > & objImg, std::vector< double > & feature, int attnShiftNum) [virtual]`

Object recognition.

Definition at line 106 of file InferenceTemporal.C.

References `computeVisualObject()`, `VisualObjectMatch::getMatchCount()`, `VisualObjectMatch::getNumberKeyCompare()`, `VisualObjectMatch::getObjectname()`, `VisualObjectMatch::getScore()`, `Timer::getSecs()`, `ModelParam< int >::getVal()`, `ModelParam< bool >::getVal()`, `luminance()`, `VisualObjectDB::matchObject()`, `VisualObjectDB::save()`, `VisualObject::setFeature()`, `VisualObject::setImage()`, `ModelManager::textLog()`, `VisualObjectDB::train()`, and `VisualObjectMatch::writeOutput()`.

**10.130.3.6** `void InferenceTemporal::start1 () [protected, virtual]`

(re-)initialization

Reimplemented from [ModelComponent](#).

Definition at line 68 of file InferenceTemporal.C.

The documentation for this class was generated from the following files:

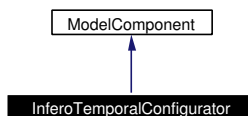
- [InferenceTemporal.H](#)
- [InferenceTemporal.C](#)

## 10.131 InferoTemporalConfigurator Class Reference

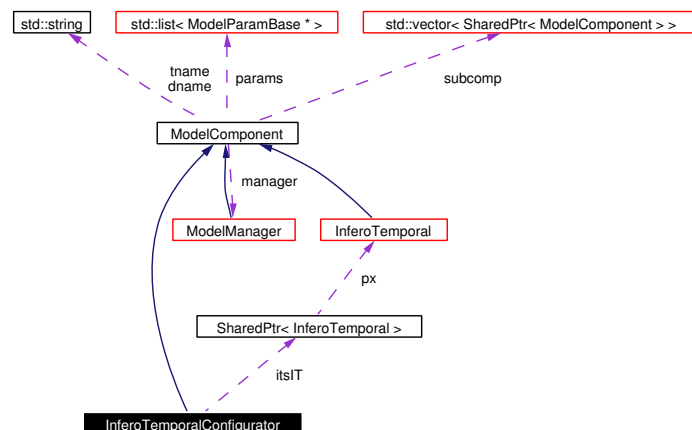
[InferoTemporal](#) configurator.

```
#include <InferoTemporalConfigurator.H>
```

Inheritance diagram for InferoTemporalConfigurator:



Collaboration diagram for InferoTemporalConfigurator:



### Public Member Functions

- [InferoTemporalConfigurator](#) ([ModelManager](#) &mgr, const [std::string](#) &descr-Name="Inferior Temporal Cortex Configurator", const [std::string](#) &tag-Name="InferoTemporalConfigurator")

*Constructor.*

- virtual [~InferoTemporalConfigurator](#) ()

*destructor*

- virtual void [exportOptions](#) (const int whatToExport, const bool recurse=true)

*Export some of our [ModelParam](#) parameters as command-line-options.*

- `SharedPtr< InferoTemporal > getIT () const`

*Get the chosen [InferoTemporal](#).*

## Protected Member Functions

- virtual bool `setModelParamString (const std::string &name, const std::string &value, const bool recurse=false, const bool warn=true)`

*Intercept people changing our [ModelParam](#).*

## Protected Attributes

- `ModelParam< std::string > itsITtype`

*type of inferior temporal cortex*

### 10.131.1 Detailed Description

[InferoTemporal](#) configurator.

This will export the `-vc-type=XX` command-line option and will instantiate a [InferoTemporal](#) of the desired type as the option gets assigned a value. As this happens, new options may become available in the command-line. To see them, use `-help` AFTER you have chosen the type to use. The current [InferoTemporal](#) may be retrieved using `getIT()`.

Definition at line 49 of file `InferoTemporalConfigurator.H`.

### 10.131.2 Constructor & Destructor Documentation

- 10.131.2.1 InferoTemporalConfigurator::InferoTemporalConfigurator**  
**([ModelManager](#) & mgr, const std::string & descrName = "Inferior Temporal Cortex Configurator", const std::string & tagName = "InferoTemporalConfigurator")**

Constructor.

Definition at line 46 of file `InferoTemporalConfigurator.C`.

References `OPTEXP_NONE`.

### 10.131.2.2 InferoTemporalConfigurator::~~InferoTemporalConfigurator () [virtual]

destructor

Definition at line 55 of file InferoTemporalConfigurator.C.

## 10.131.3 Member Function Documentation

### 10.131.3.1 void InferoTemporalConfigurator::exportOptions (const int *whatToExport*, const bool *recurse* = true) [virtual]

Export some of our [ModelParam](#) parameters as command-line-options.

See the base function in [ModelComponent.H](#) for info. Actually, here we just keep a copy of *whatToExport* and we will call [exportOptions\(\)](#) on our [InferoTemporal](#) once we know its type and instantiate it.

Reimplemented from [ModelComponent](#).

Definition at line 59 of file InferoTemporalConfigurator.C.

References [ModelComponent::exportOptions\(\)](#), [itsITtype](#), [OPTEXP\\_CORE](#), and [ModelManager::requestOption\(\)](#).

### 10.131.3.2 [SharedPtr](#)< [InferoTemporal](#) > InferoTemporalConfigurator::getIT () const

Get the chosen [InferoTemporal](#).

You should call this during [start\(\)](#) of the [ModelComponent](#) that needs the [InferoTemporal](#). CAUTION: this may be a [SharedPtr\(NULL\)](#) if no controller has been chosen.

Definition at line 77 of file InferoTemporalConfigurator.C.

### 10.131.3.3 bool InferoTemporalConfigurator::setModelParamString (const [std::string](#) & *name*, const [std::string](#) & *value*, const bool *recurse* = false, const bool *warn* = true) [protected, virtual]

Intercept people changing our [ModelParam](#).

See [ModelComponent.H](#); as parsing the command-line or reading a config file sets our name, we'll also here instantiate a controller of the proper type (and export its options)

Reimplemented from [ModelComponent](#).

Definition at line 82 of file InferoTemporalConfigurator.C.

References `ModelComponent::addSubComponent()`, `SharedPtr< IneroTemporal >::get()`, `itsITtype`, `ModelComponent::removeSubComponent()`, `SharedPtr< IneroTemporal >::reset()`, and `ModelComponent::setModelParamString()`.

## 10.131.4 Member Data Documentation

### 10.131.4.1 `ModelParam<std::string> IneroTemporalConfigurator::itsITtype` [protected]

type of inferior temporal cortex

Definition at line 77 of file `IneroTemporalConfigurator.H`.

Referenced by `exportOptions()`, and `setModelParamString()`.

The documentation for this class was generated from the following files:

- [IneroTemporalConfigurator.H](#)
- [IneroTemporalConfigurator.C](#)

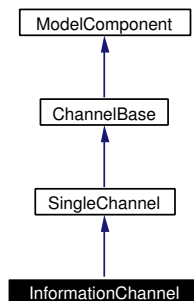


## 10.132 InformationChannel Class Reference

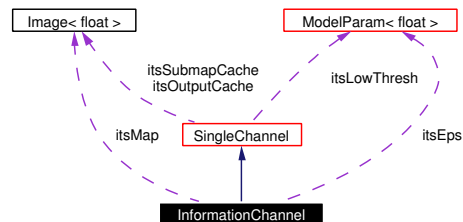
An information channel.

```
#include <Channels.H>
```

Inheritance diagram for InformationChannel:



Collaboration diagram for InformationChannel:



### Public Member Functions

- [InformationChannel](#) ([ModelManager](#) &mgr)  
*Construct with standard params.*
- virtual [~InformationChannel](#) ()  
*Destructor.*
- virtual bool [outputAvailable](#) () const  
*overload*

## Protected Member Functions

- virtual void `doInput` (const `Image< PixRGB< float > > *`, const `Image< float > *bwimg`, const double `t`, const `Image< byte > &clipMask`)  
*special input function*
- virtual `Image< float > getOutput ()`  
*special output function*

## Protected Attributes

- `ModelParam< float > itsEps`  
*epsilon on contrast sensitivity threshold*
- `Image< float > itsMap`  
*our information map*

### 10.132.1 Detailed Description

An information channel.

CAUTION: this channel will throw a fatal exception if your `LevelSpec` contains more than one level or if that level is different from the `maplevel` or if the `deltas` are non-zero.

Definition at line 614 of file `Channels.H`.

### 10.132.2 Constructor & Destructor Documentation

#### 10.132.2.1 `InformationChannel::InformationChannel (ModelManager & mgr)`

Construct with standard params.

Definition at line 2849 of file `Channels.C`.

References `INFORMATION`.

#### 10.132.2.2 `InformationChannel::~~InformationChannel ()` [virtual]

Destructor.

Definition at line 2857 of file `Channels.C`.

### 10.132.3 Member Function Documentation

**10.132.3.1** `void InformationChannel::doInput (const Image< PixRGB< float > > *, const Image< float > * bwimg, const double t, const Image< byte > & clipMask)` [protected, virtual]

special input function

Reimplemented from [SingleChannel](#).

Definition at line 2865 of file Channels.C.

References `ASSERT`, `LevelSpec::delMax()`, `LevelSpec::delMin()`, `ModelParam< float >::getVal()`, `ModelParam< LevelSpec >::getVal()`, `infoMeasure()`, `itsEps`, `LevelSpec::levMax()`, `LevelSpec::levMin()`, `LevelSpec::mapLevel()`, and `uint`.

**10.132.3.2** `Image< float > InformationChannel::getOutput ()` [protected, virtual]

special output function

Reimplemented from [SingleChannel](#).

Definition at line 2883 of file Channels.C.

**10.132.3.3** `bool InformationChannel::outputAvailable () const` [virtual]

overload

Reimplemented from [SingleChannel](#).

Definition at line 2861 of file Channels.C.

References `Image< float >::initialized()`.

### 10.132.4 Member Data Documentation

**10.132.4.1** `ModelParam<float> InformationChannel::itsEps` [protected]

epsilon on contrast sensitivity threshold

Definition at line 627 of file Channels.H.

Referenced by `doInput()`.

**10.132.4.2** `Image<float> InformationChannel::itsMap` [protected]

our information map

Definition at line 637 of file Channels.H.

The documentation for this class was generated from the following files:

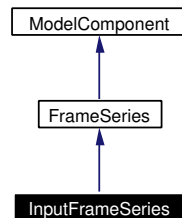
- [Channels.H](#)
- [Channels.C](#)

## 10.133 InputFrameSeries Class Reference

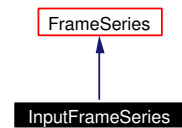
A [FrameSeries](#) with additional facilities for reading input frames.

```
#include <FrameSeries.H>
```

Inheritance diagram for InputFrameSeries:



Collaboration diagram for InputFrameSeries:



### Public Member Functions

- [InputFrameSeries](#) ([ModelManager](#) &mgr, const [std::string](#) &descrName="Frame Series", const [std::string](#) &tagName="FrameSeries")

*Constructor.*

- virtual [~InputFrameSeries](#) ()

*Destructor.*

- [Dims getImageDims](#) (const [RasterFileFormat](#) ft, const int framenum, const [Raster::ColorMode](#) cmode, const char \*otherstem,...) const

*Get dims of a frame on disk, without fully reading it.*

- [Dims getImageDims](#) (const [RasterFileFormat](#) ft=Auto, const int framenum=-1, const [Raster::ColorMode](#) cmode=[Raster::MODE\\_RGB](#)) const

*Get dims of a frame on disk, without fully reading it.*

- `Image< PixRGB< byte > > readRGB` (const `RasterFileFormat` ft, const int framenum, const char \*otherstem,...)

*Read a frame from disk.*

- `Image< PixRGB< byte > > readRGB` (const `RasterFileFormat` ft=Auto, const int framenum=-1)

*Read a frame from disk, using default filename stem.*

- `Image< byte > readGray` (const `RasterFileFormat` ft, const int framenum, const char \*otherstem,...)

*Read a greyscale frame.*

- `Image< byte > readGray` (const `RasterFileFormat` ft=Auto, const int framenum=-1)

*Read a greyscale frame.*

## Protected Member Functions

- virtual void `start2` ()

*post-commandline-parsing initialization (see [ModelComponent.H](#))*

### 10.133.1 Detailed Description

A `FrameSeries` with additional facilities for reading input frames.

NOTE: when `start()` is called, we will set the "FOAradius", "FoveaRadius", "Input-FrameDims", "SCinitialEyePosition" and "SCinitialHeadPosition" model parameters from input image dims, if someone wants them and they have not been set.

Definition at line 185 of file `FrameSeries.H`.

### 10.133.2 Constructor & Destructor Documentation

#### 10.133.2.1 `InputFrameSeries::InputFrameSeries` (`ModelManager` & mgr, const `std::string` & descrName = "Frame Series", const `std::string` & tagName = "FrameSeries")

Constructor.

Definition at line 286 of file `FrameSeries.C`.

**10.133.2.2 InputFrameSeries::~~InputFrameSeries () [virtual]**

Destructor.

Definition at line 293 of file FrameSeries.C.

**10.133.3 Member Function Documentation****10.133.3.1 Dims InputFrameSeries::getImageDims (const RasterFileFormat *ft* = Auto, const int *framenum* = -1, const Raster::ColorMode *cmode* = Raster::MODE\_RGB) const**

Get dims of a frame on disk, without fully reading it.

It is okay to call this before the model is [started\(\)](#). If input resizing is being done, this will return the resized dims. All in all, this will return the size of whatever you will get when calling [readRGB\(\)](#) or [readGray\(\)](#).

Definition at line 387 of file FrameSeries.C.

References [getImageDims\(\)](#).

**10.133.3.2 Dims InputFrameSeries::getImageDims (const RasterFileFormat *ft*, const int *framenum*, const Raster::ColorMode *cmode*, const char \* *otherstem*, ...) const**

Get dims of a frame on disk, without fully reading it.

It is okay to call this before the model is [started\(\)](#). If input resizing is being done, this will return the resized dims. All in all, this will return the size of whatever you will get when calling [readRGB\(\)](#) or [readGray\(\)](#).

Definition at line 360 of file FrameSeries.C.

References [FrameSeries::computeFileName\(\)](#), [FrameRange::first](#), [Raster::getImageDims\(\)](#), [ModelParam< RasterFileFormat >::getVal\(\)](#), [ModelParam< Dims >::getVal\(\)](#), [ModelParam< FrameRange >::getVal\(\)](#), [Dims::h\(\)](#), [FrameRange::isMultiframe\(\)](#), [ModelComponent::started\(\)](#), and [Dims::w\(\)](#).

Referenced by [getImageDims\(\)](#), and [start2\(\)](#).

**10.133.3.3 Image< byte > InputFrameSeries::readGray (const RasterFileFormat *ft* = Auto, const int *framenum* = -1)**

Read a greyscale frame.

Definition at line 449 of file FrameSeries.C.

References [readGray\(\)](#).

#### 10.133.3.4 **Image< byte > InputFrameSeries::readGray (const RasterFileFormat *ft*, const int *framenum*, const char \* *otherstem*, ...)**

Read a greyscale frame.

Definition at line 424 of file FrameSeries.C.

References `FrameSeries::computeFileName()`, `Raster::Display()`, `ModelParam< RasterFileFormat >::getVal()`, `ModelParam< FrameRange >::getVal()`, `FrameRange::isMultiframe()`, `FrameSeries::okToDisplay()`, `RasterFileFormat`, `FrameSeries::resizeImage()`, and `ModelManager::textLog()`.

Referenced by `readGray()`.

#### 10.133.3.5 **Image< PixRGB< byte > > InputFrameSeries::readRGB (const RasterFileFormat *ft* = Auto, const int *framenum* = -1)**

Read a frame from disk, using default filename stem.

The default is to read using the stem set by `setFileStem()`, and the current frame number (if multiframe)

Definition at line 419 of file FrameSeries.C.

References `readRGB()`.

#### 10.133.3.6 **Image< PixRGB< byte > > InputFrameSeries::readRGB (const RasterFileFormat *ft*, const int *framenum*, const char \* *otherstem*, ...)**

Read a frame from disk.

##### Parameters:

***ft*** `RasterFileFormat` as supported by `Raster`

***framenum*** frame number, or use -1 for the current frame.

***otherstem*** prefix stem that may contain printf-like fields and will be used with any additional arguments. If multiframe, the filename will be of the form , otherwise it will be

Definition at line 394 of file FrameSeries.C.

References `FrameSeries::computeFileName()`, `Raster::Display()`, `ModelParam< RasterFileFormat >::getVal()`, `ModelParam< FrameRange >::getVal()`, `FrameRange::isMultiframe()`, `FrameSeries::okToDisplay()`, `RasterFileFormat`, `FrameSeries::resizeImage()`, and `ModelManager::textLog()`.

Referenced by `readRGB()`.



### 10.133.3.7 void InputFrameSeries::start2 () [protected, virtual]

post-commandline-parsing initialization (see [ModelComponent.H](#))

Will peek the first frame (without fully loading it) and set "FOAradius", "FoveaRadius", "InputFrameDims" "SCinitialEyePosition" and "SCinitialHeadPosition" model parameters from input image dims if someone wants them and they have not been set

Reimplemented from [FrameSeries](#).

Definition at line 297 of file FrameSeries.C.

References [getImageDims\(\)](#), [ModelComponent::getModelParamVal\(\)](#), [Dims::h\(\)](#), [Point2D::i](#), [Dims::isEmpty\(\)](#), [Point2D::j](#), [ModelComponent::setModelParamVal\(\)](#), [FrameSeries::start2\(\)](#), [ModelManager::textLog\(\)](#), and [Dims::w\(\)](#).

The documentation for this class was generated from the following files:

- [FrameSeries.H](#)
- [FrameSeries.C](#)

## 10.134 IntensityChannel Class Reference

An intensity channel.

```
#include <Channels.H>
```

Inheritance diagram for IntensityChannel: Collaboration diagram for IntensityChannel:



### Public Member Functions

- [IntensityChannel](#) ([ModelManager](#) &mgr)  
*Construct with standard params.*
- virtual [~IntensityChannel](#) ()  
*Destructor.*

#### 10.134.1 Detailed Description

An intensity channel.

Definition at line 60 of file Channels.H.

#### 10.134.2 Constructor & Destructor Documentation

##### 10.134.2.1 IntensityChannel::IntensityChannel ([ModelManager](#) & mgr)

Construct with standard params.

Definition at line 76 of file Channels.C.

References `INTENS`, `makeShared()`, and `ModelParam< bool >::setVal()`.

##### 10.134.2.2 IntensityChannel::~~IntensityChannel () [virtual]

Destructor.

Definition at line 84 of file Channels.C.

The documentation for this class was generated from the following files:

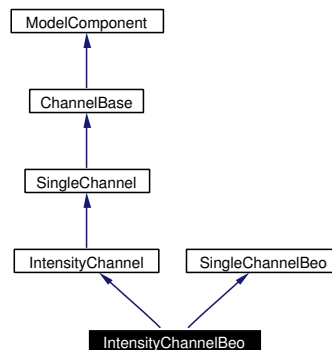
- [Channels.H](#)
- [Channels.C](#)

## 10.135 IntensityChannelBeo Class Reference

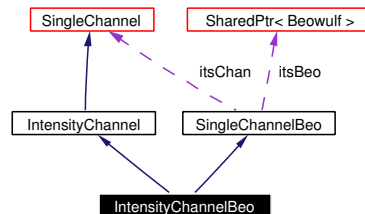
Beowulf version of [IntensityChannel](#).

```
#include <ChannelsBeo.H>
```

Inheritance diagram for IntensityChannelBeo:



Collaboration diagram for IntensityChannelBeo:



### Public Member Functions

- **IntensityChannelBeo** ([ModelManager](#) &mgr, const [SharedPtr< Beowulf >](#) &beow)
- virtual bool [outputAvailable](#) () const  
*Query whether the channel is ready to give yield valid output.*

### Protected Member Functions

- virtual void [doInput](#) (const [Image< PixRGB< float > >](#) \*, const [Image< float >](#) \*bwimg, const double t, const [Image< byte >](#) &clipMask)

*SingleChannel* requires only luminance input.

- virtual void [start1](#) ()

### 10.135.1 Detailed Description

[Beowulf](#) version of [IntensityChannel](#).

Definition at line 52 of file ChannelsBeo.H.

### 10.135.2 Member Function Documentation

**10.135.2.1** void [IntensityChannelBeo::doInput](#) (const [Image](#)< [PixRGB](#)< float > > \*, const [Image](#)< float > \* *bwimg*, const double *t*, const [Image](#)< byte > & *clipMask*) [protected, virtual]

[SingleChannel](#) requires only luminance input.

Reimplemented from [SingleChannel](#).

Definition at line 63 of file ChannelsBeo.C.

References [SingleChannelBeo::sendInput](#)().

**10.135.2.2** bool [IntensityChannelBeo::outputAvailable](#) () const [virtual]

Query whether the channel is ready to give yield valid output.

In some cases (e.g. with motion channels), it may take several input frames before the channel is fully initialized.

Reimplemented from [SingleChannel](#).

Definition at line 56 of file ChannelsBeo.C.

References [SingleChannel::outputAvailable](#)().

**10.135.2.3** void [IntensityChannelBeo::start1](#) () [protected, virtual]

If you overload this, make sure you call [SingleChannel::start1\(\)](#) at the beginning of your overload

Reimplemented from [SingleChannel](#).

Definition at line 67 of file ChannelsBeo.C.

References [SingleChannelBeo::requestBeowulfNode](#)(), and [SingleChannel::start1](#)().

The documentation for this class was generated from the following files:

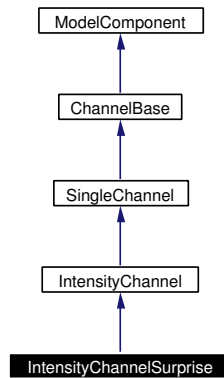
- [ChannelsBeo.H](#)
- [ChannelsBeo.C](#)

## 10.136 IntensityChannelSurprise Class Reference

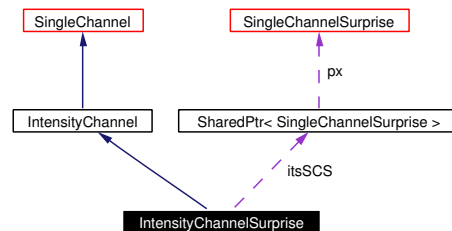
An intensity channel.

```
#include <ChannelsSurprise.H>
```

Inheritance diagram for IntensityChannelSurprise:



Collaboration diagram for IntensityChannelSurprise:



### Public Member Functions

- [IntensityChannelSurprise](#) ([ModelManager](#) &mgr)  
*Construct with standard params.*
- virtual [~IntensityChannelSurprise](#) ()  
*Destructor.*
- virtual [Image< float > computeSubmap](#) (const [uint](#) index) const  
*[computeSubmap\(\)](#) is special and will return surprise*

## Protected Attributes

- [SharedPtr](#)< [SingleChannelSurprise](#) > [itsSCS](#)  
*our surprise computer*

### 10.136.1 Detailed Description

An intensity channel.

Definition at line 51 of file ChannelsSurprise.H.

### 10.136.2 Constructor & Destructor Documentation

#### 10.136.2.1 [IntensityChannelSurprise::IntensityChannelSurprise](#) ([ModelManager](#) & *mgr*)

Construct with standard params.

Definition at line 64 of file ChannelsSurprise.C.

References [ModelComponent::addSubComponent\(\)](#), and [itsSCS](#).

#### 10.136.2.2 [IntensityChannelSurprise::~~IntensityChannelSurprise](#) () [virtual]

Destructor.

Definition at line 70 of file ChannelsSurprise.C.

### 10.136.3 Member Function Documentation

#### 10.136.3.1 [Image](#)< float > [IntensityChannelSurprise::computeSubmap](#) (const [uint](#) *index*) const [virtual]

[computeSubmap\(\)](#) is special and will return surprise

Reimplemented from [SingleChannel](#).

Definition at line 73 of file ChannelsSurprise.C.

References [uint](#).



## 10.136.4 Member Data Documentation

### 10.136.4.1 [SharedPtr<SingleChannelSurprise>](#) [IntensityChannelSurprise::its-SCS](#) [protected]

our surprise computer

Definition at line 64 of file ChannelsSurprise.H.

Referenced by IntensityChannelSurprise().

The documentation for this class was generated from the following files:

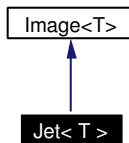
- [ChannelsSurprise.H](#)
- ChannelsSurprise.C

## 10.137 Jet< T > Class Template Reference

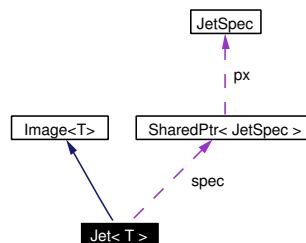
This class specifies a Jet, i.e., array of feature values for one location.

```
#include <Jet.H>
```

Inheritance diagram for Jet< T >:



Collaboration diagram for Jet< T >:



## Public Member Functions

### Constructors, initialization and destructor

- **Jet** (const **SharedPtr**< **JetSpec** > js)  
*Constructor; must provide a fully-configured JetSpec.*
- **Jet** ()  
*Uninitialized constructor (for arrays of Jets).*
- void **init** (const **SharedPtr**< **JetSpec** > js)  
*Initialization of an uninitialized Jet.*
- **Jet**< T > & **operator=** (const **Image**< T > &img)  
*Load from an Image, preserving our JetSpec.*
- template<class TT> **Jet**< T > & **operator=** (const **Image**< TT > &img)  
*Load from an image, preserving our JetSpec and doing a clamped convert.*

- `~Jet ()`  
*Destructor.*

### Access functions

- `T getVal (const VisualFeature f, const VisualFeatureType t,...) const`  
*Get a value, for given feature, type and indices.*
- `T getValV (const VisualFeature f, const VisualFeatureType t, const std::vector< int > &v) const`  
*Get a value, for given feature, type and vector of indices.*
- `T getVal (const int idx) const`  
*Get a value by absolute index in the Jet array. Use with caution!*
- `void setVal (const T &val, const VisualFeature f, const VisualFeatureType t,...)`  
*Set a value, for given feature, type and indices.*
- `void setValV (const T &val, const VisualFeature f, const VisualFeatureType t, const std::vector< int > &v)`  
*Set a value, for given feature, type and vector of indices.*
- `void setVal (const int idx, const T &val)`  
*Set a value by absolute index in the Jet array. Use with caution!*
- `SharedPtr< JetSpec > getSpec () const`  
*Retrieve the JetSpec in case your original has run out of scope.*

### Convenience JetSpec Access Methods

- `bool hasFeature (const VisualFeature f) const`  
*Query whether we have some representation for given feature.*
- `bool hasFeatureType (const VisualFeature f, const VisualFeatureType t) const`  
*Query whether we have some representation for given feature/type.*
- `int getIndexBase (const VisualFeature f, const VisualFeatureType t, int &siz) const`  
*Get index of first Jet element for given feature & type.*
- `int getNbIndexRanges (const VisualFeature f, const VisualFeatureType t) const`

*Get number of index ranges.*

- bool [getIndexRange](#) (const [VisualFeature](#) f, const [VisualFeatureType](#) t, const int idxnum, int &idxmin, int &idxmax) const

*Get an index range; true if range exists in [JetSpec](#).*

### 10.137.1 Detailed Description

**template<class T> class Jet< T >**

This class specifies a Jet, i.e., array of feature values for one location.

The features represented in the Jet are defined by a [JetSpec](#) object that must be passed at construction of the Jet. Jet extends [Image](#) and inherits all of its functionality. The [JetSpec](#) is passed as a [SharedPtr](#), so that the original can run out of scope and be destroyed (and the actual data will be freed upon destruction of the last Jet built from that [JetSpec](#)).

Definition at line 180 of file Jet.H.

### 10.137.2 Constructor & Destructor Documentation

**10.137.2.1 template<class T> [Jet](#)< T >::[Jet](#) (const [SharedPtr](#)< [JetSpec](#) > js)**

Constructor; must provide a fully-configured [JetSpec](#).

Definition at line 230 of file Jet.C.

References [Image](#)< T >::resize().

**10.137.2.2 template<class T> [Jet](#)< T >::[Jet](#) ()**

Uninitialized constructor (for arrays of Jets).

Definition at line 235 of file Jet.C.

**10.137.2.3 template<class T> [Jet](#)< T >::~~[Jet](#) ()**

Destructor.

Definition at line 245 of file Jet.C.

References [Image](#)< T >::freeMem().

### 10.137.3 Member Function Documentation

**10.137.3.1** `template<class T> int Jet< T >::getIndexBase (const VisualFeature f, const VisualFeatureType t, int & siz) const [inline]`

Get index of first Jet element for given feature & type.

Also returns total size allocated in the Jet for this feature/type

Definition at line 504 of file Jet.H.

**10.137.3.2** `template<class T> bool Jet< T >::getIndexRange (const VisualFeature f, const VisualFeatureType t, const int idxnum, int & idxmin, int & idxmax) const [inline]`

Get an index range; true if range exists in JetSpec.

Definition at line 518 of file Jet.H.

Referenced by SingleChannel::fillJet(), SingleChannel::fillJetNI(), and raodistance().

**10.137.3.3** `template<class T> int Jet< T >::getNbIndexRanges (const VisualFeature f, const VisualFeatureType t) const [inline]`

Get number of index ranges.

Definition at line 511 of file Jet.H.

Referenced by raodistance().

**10.137.3.4** `template<class T> SharedPtr< JetSpec > Jet< T >::getSpec () const [inline]`

Retrieve the JetSpec in case your original has run out of scope.

Definition at line 488 of file Jet.H.

Referenced by SingleChannel::fillJet(), SingleChannel::fillJetNI(), VisualCortex::get-JetDistMap(), and VisualCortex::getRaoJetMap().

**10.137.3.5** `template<class T> T Jet< T >::getVal (const int idx) const [inline]`

Get a value by absolute index in the Jet array. Use with caution!

Reimplemented from Image< T >.

Definition at line 460 of file Jet.H.

References Image< T >::begin().

**10.137.3.6** `template<class T> T Jet< T >::getVal (const VisualFeature f, const VisualFeatureType t, ...) const [inline]`

Get a value, for given feature, type and indices.

Definition at line 442 of file Jet.H.

References Image< T >::getVal().

Referenced by raodistance().

**10.137.3.7** `template<class T> T Jet< T >::getValV (const VisualFeature f, const VisualFeatureType t, const std::vector< int > & v) const [inline]`

Get a value, for given feature, type and vector of indices.

Definition at line 451 of file Jet.H.

References Image< T >::getVal().

**10.137.3.8** `template<class T> bool Jet< T >::hasFeature (const VisualFeature f) const [inline]`

Query whether we have some representation for given feature.

Definition at line 493 of file Jet.H.

**10.137.3.9** `template<class T> bool Jet< T >::hasFeatureType (const VisualFeature f, const VisualFeatureType t) const [inline]`

Query whether we have some representation for given feature/type.

Definition at line 498 of file Jet.H.

Referenced by raodistance().

**10.137.3.10** `template<class T> void Jet< T >::init (const SharedPtr< JetSpec > js)`

Initialization of an uninitialized Jet.

Definition at line 240 of file Jet.C.

References ASSERT, SharedPtr< JetSpec >::get(), and Image< T >::resize().

**10.137.3.11** `template<class T> template<class TT> Jet< T > & Jet< T >::operator= (const Image< TT > & img) [inline]`

Load from an image, preserving our JetSpec and doing a clamped convert.

The Jet must have been initialized previously and its size must match that of the Image. This is potentially dangerous, and mostly provided so that we can inherit all of the Image free functions that return images (e.g., those of Image\_MathOps.H). This is an expensive operation in which we convert each pixel in turn.

Definition at line 434 of file Jet.H.

References ASSERT, Image< T >::isSameSize(), and Image< T >::operator=().

**10.137.3.12** `template<class T> Jet< T > & Jet< T >::operator= (const Image< T > & img) [inline]`

Load from an Image, preserving our JetSpec.

The Jet must have been initialized previously and its size must match that of the Image. This is potentially dangerous, and mostly provided so that we can inherit all of the Image free functions that return images (e.g., those of Image\_MathOps.H). This is a cheap operation thanks to the copy-on-write/ref-counting

Reimplemented from Image< T >.

Definition at line 424 of file Jet.H.

References ASSERT, Image< T >::isSameSize(), and Image< T >::swap().

**10.137.3.13** `template<class T> void Jet< T >::setVal (const int idx, const T & val) [inline]`

Set a value by absolute index in the Jet array. Use with caution!

Definition at line 483 of file Jet.H.

References Image< T >::beginw().

**10.137.3.14** `template<class T> void Jet< T >::setVal (const T & val, const VisualFeature f, const VisualFeatureType t, ...) [inline]`

Set a value, for given feature, type and indices.

Definition at line 465 of file Jet.H.

Referenced by SingleChannel::fillJet(), SingleChannel::fillJetNI(), and Jet< T >::setValV().

**10.137.3.15** `template<class T> void Jet< T >::setValV (const T & val, const VisualFeature f, const VisualFeatureType t, const std::vector< int > & v) [inline]`

Set a value, for given feature, type and vector of indices.

Definition at line 475 of file Jet.H.

References Jet< T >::setVal().

The documentation for this class was generated from the following files:

- [Jet.H](#)
- [Jet.C](#)



## 10.138 JetSpec Class Reference

This class specifies the features, scales and indices represented in a [Jet](#).

```
#include <Jet.H>
```

### Public Member Functions

#### Constructors, destructor and operators

- [JetSpec](#) ()  
*Empty constructor; starts with an empty specification.*
- [JetSpec](#) (const [JetSpec](#) &js)  
*Copy constructor.*
- [JetSpec](#) & operator= (const [JetSpec](#) &js)  
*Assignment operator.*
- void [freeMem](#) ()  
*Free all allocated memory and un-initialize.*
- ~[JetSpec](#) ()  
*Destructor.*

#### Build methods

- bool [addFeature](#) (const [VisualFeature](#) f)  
*Add a new feature; returns true on success.*
- bool [addFeatureType](#) (const [VisualFeature](#) f, const [VisualFeatureType](#) t)  
*Add a new representation type for an existing feature; true on success.*
- bool [addIndexRange](#) (const [VisualFeature](#) f, const [VisualFeatureType](#) t, const int indexmin, const int indexmax)  
*Add a new index range for existing feature and type; true on success.*

#### Access methods

- int [getJetSize](#) () const  
*Get the number of components in a jet with our specs.*
- bool [hasFeature](#) (const [VisualFeature](#) f) const  
*Query whether we have some representation for given feature.*

- bool `hasFeatureType` (const `VisualFeature` f, const `VisualFeatureType` t) const  
*Query whether we have some representation for given feature/type.*
- int `getIndexBase` (const `VisualFeature` f, const `VisualFeatureType` t, int &sz) const  
*Get index of first `Jet` element for given feature & type.*
- int `getNbIndexRanges` (const `VisualFeature` f, const `VisualFeatureType` t) const  
*Get number of index ranges.*
- bool `getIndexRange` (const `VisualFeature` f, const `VisualFeatureType` t, const int idxnum, int &idxmin, int &idxmax) const  
*Get an index range; true if range exists in `JetSpec`.*
- void `print` () const  
*Debugging printout to stdout.*

#### Jet data access methods

- int `getIndex` (const `VisualFeature` f, const `VisualFeatureType` t,...) const  
*Get index in `Jet` array data for given feature, type and set of indices.*
- int `getIndexA` (const `VisualFeature` f, const `VisualFeatureType` t, va\_list &a) const  
*`getIndex()` with `va_list` parameters*
- int `getIndexV` (const `VisualFeature` f, const `VisualFeatureType` t, const `std::vector`< int > &v) const  
*`getIndex()` with vector of parameters*

### 10.138.1 Detailed Description

This class specifies the features, scales and indices represented in a `Jet`.

A `Jet` object requires a valid `JetSpec` object at construction

Definition at line 53 of file `Jet.H`.

## 10.138.2 Constructor & Destructor Documentation

### 10.138.2.1 JetSpec::JetSpec ()

Empty constructor; starts with an empty specification.

Definition at line 43 of file Jet.C.

References `freeMem()`.

### 10.138.2.2 JetSpec::JetSpec (const JetSpec & js)

Copy constructor.

Definition at line 47 of file Jet.C.

References `freeMem()`.

### 10.138.2.3 JetSpec::~~JetSpec ()

Destructor.

Definition at line 91 of file Jet.C.

References `freeMem()`.

## 10.138.3 Member Function Documentation

### 10.138.3.1 bool JetSpec::addFeature (const VisualFeature f)

Add a new feature; returns true on success.

Definition at line 95 of file Jet.C.

References `featureName()`.

Referenced by `addFeatureType()`, `addIndexRange()`, and `operator=()`.

### 10.138.3.2 bool JetSpec::addFeatureType (const VisualFeature f, const VisualFeatureType t)

Add a new representation type for an existing feature; true on success.

Definition at line 112 of file Jet.C.

References `addFeature()`, `featureName()`, `featureTypeName()`, `hasFeature()`, and `hasFeatureType()`.

Referenced by `addIndexRange()`, and `operator=()`.

### 10.138.3.3 **bool JetSpec::addIndexRange** (const **VisualFeature** *f*, const **VisualFeatureType** *t*, const int *indexmin*, const int *indexmax*)

Add a new index range for existing feature and type; true on success.

Valid indices will be in [*indexmin*..*indexmax*] inclusive on both sides.

Definition at line 151 of file Jet.C.

References `addFeature()`, `addFeatureType()`, `hasFeature()`, and `hasFeatureType()`.

Referenced by `SingleChannel::fillJetSpec()`, `ComplexChannel::fillJetSpec()`, `BeobotVisualCortex::init()`, and `operator=()`.

### 10.138.3.4 **void JetSpec::freeMem** ()

Free all allocated memory and un-initialize.

Definition at line 73 of file Jet.C.

References `NBVISUALFEATURES`.

Referenced by `JetSpec()`, `operator=()`, and `~JetSpec()`.

### 10.138.3.5 **int JetSpec::getIndex** (const **VisualFeature** *f*, const **VisualFeatureType** *t*, ...) const [inline]

Get index in `Jet` array data for given feature, type and set of indices.

The ... arguments should be as many ints as we have have `addIndexRange()`'d when building the `JetSpec`, and their values should be within the ranges specified at the time of `addIndexRange()`. Returns -1 if feature or type unknown or if any of the indices is out of range.

Definition at line 344 of file Jet.H.

References `getIndexA()`.

### 10.138.3.6 **int JetSpec::getIndexA** (const **VisualFeature** *f*, const **VisualFeatureType** *t*, va\_list & *a*) const [inline]

`getIndex()` with `va_list` parameters

Definition at line 352 of file Jet.H.

References `featureName()`, and `featureTypeName()`.

Referenced by `getIndex()`.

**10.138.3.7** `int JetSpec::getIndexBase (const VisualFeature f, const VisualFeatureType t, int & siz) const [inline]`

Get index of first Jet element for given feature & type.

Also returns the total size allocated in the Jet for this feature and type

Definition at line 314 of file Jet.H.

References ASSERT.

**10.138.3.8** `bool JetSpec::getIndexRange (const VisualFeature f, const VisualFeatureType t, const int idxnum, int & idxmin, int & idxmax) const [inline]`

Get an index range; true if range exists in JetSpec.

Definition at line 331 of file Jet.H.

**10.138.3.9** `int JetSpec::getIndexV (const VisualFeature f, const VisualFeatureType t, const std::vector< int > & v) const [inline]`

`getIndex()` with vector of parameters

Definition at line 379 of file Jet.H.

References `featureName()`, and `featureTypeName()`.

**10.138.3.10** `int JetSpec::getJetSize () const [inline]`

Get the number of components in a jet with our specs.

Definition at line 297 of file Jet.H.

**10.138.3.11** `int JetSpec::getNbIndexRanges (const VisualFeature f, const VisualFeatureType t) const [inline]`

Get number of index ranges.

Definition at line 323 of file Jet.H.

References ASSERT.

**10.138.3.12** `bool JetSpec::hasFeature (const VisualFeature f) const [inline]`

Query whether we have some representation for given feature.

Definition at line 301 of file Jet.H.

Referenced by `addFeatureType()`, and `addIndexRange()`.

#### **10.138.3.13** `bool JetSpec::hasFeatureType (const VisualFeature f, const VisualFeatureType t) const` `[inline]`

Query whether we have some representation for given feature/type.

Definition at line 309 of file Jet.H.

Referenced by `addFeatureType()`, and `addIndexRange()`.

#### **10.138.3.14** `JetSpec & JetSpec::operator= (const JetSpec & js)`

Assignment operator.

Definition at line 51 of file Jet.C.

References `addFeature()`, `addFeatureType()`, `addIndexRange()`, `freeMem()`, and `spec`.

#### **10.138.3.15** `void JetSpec::print () const`

Debugging printout to stdout.

Definition at line 178 of file Jet.C.

References `featureName()`, and `featureTypeName()`.

Referenced by `BeobotVisualCortex::init()`.

The documentation for this class was generated from the following files:

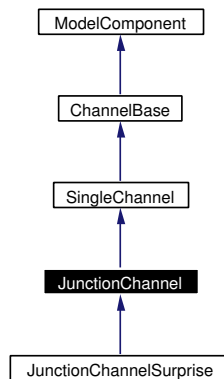
- [Jet.H](#)
- [Jet.C](#)

## 10.139 JunctionChannel Class Reference

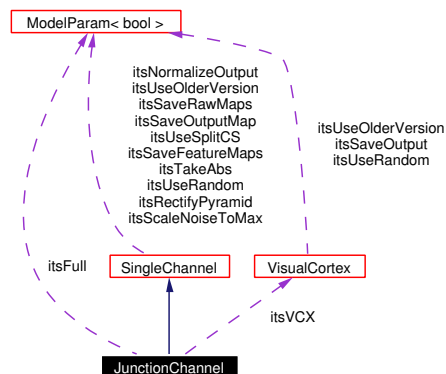
A junction channel.

```
#include <Channels.H>
```

Inheritance diagram for JunctionChannel:



Collaboration diagram for JunctionChannel:



### Public Member Functions

- `JunctionChannel` (`ModelManager` &mgr, `VisualCortex` \*vcx, const `VisualFeature` vs=UNKNOWN, const int R0=0, const int R1=0, const int R2=0, const int R3=0, const int R4=0, const int R5=0, const int R6=0, const int R7=0)
- virtual `~JunctionChannel` ()

*Destructor.*

- virtual void [exportOptions](#) (const int whatToExport, const bool recurse=true)

*Export some of our [ModelParam](#) parameters as command-line-options.*

## Protected Member Functions

- virtual void [doInput](#) (const [Image](#)< [PixRGB](#)< float > > \*, const [Image](#)< float > \*bwimg, const double t, const [Image](#)< byte > &clipMask)

*Computes the corners from the gabor channels accessed via the.*

## Protected Attributes

- [ModelParam](#)< bool > [itsFull](#)
- [VisualCortex](#) \* [itsVCX](#)

*keep track of our [VisualCortex](#)*

### 10.139.1 Detailed Description

A junction channel.

Definition at line 423 of file Channels.H.

### 10.139.2 Constructor & Destructor Documentation

- 10.139.2.1 JunctionChannel::JunctionChannel ([ModelManager](#) & mgr, [VisualCortex](#) \* vcx, const [VisualFeature](#) vs = UNKNOWN, const int R0 = 0, const int R1 = 0, const int R2 = 0, const int R3 = 0, const int R4 = 0, const int R5 = 0, const int R6 = 0, const int R7 = 0)**

Construct with the manager, visual cortex, visual feature and relevance of the different oriented features (0 => 0 deg., 1 => 45 deg., ... , 7 => 315 deg., and whether we want a full or partial implementation of the junction

Definition at line 1678 of file Channels.C.

References [ENDPOINT](#), [LJUNCTION](#), [ModelComponent::setDescriptiveName\(\)](#), [ModelComponent::setTagName\(\)](#), [ModelParam](#)< bool >::[setVal\(\)](#), [ChannelBase::setVisualFeature\(\)](#), [TJUNCTION](#), and [XJUNCTION](#).



**10.139.2.2 JunctionChannel::~JunctionChannel()** [virtual]

Destructor.

Definition at line 1709 of file Channels.C.

**10.139.3 Member Function Documentation****10.139.3.1 void JunctionChannel::doInput (const Image< PixRGB< float > > \*, const Image< float > \* *bwimg*, const double *t*, const Image< byte > & *clipMask*)** [protected, virtual]

Computes the corners from the gabor channels accessed via the.

Reimplemented from [SingleChannel](#).

Definition at line 1725 of file Channels.C.

References ASSERT, VisualCortex::channel(), ModelComponent::getModelParamVal(), ModelParam< bool >::getVal(), VisualCortex::hasChannel(), itsFull, itsVCX, junctionFilterFull(), junctionFilterPartial(), SingleChannel::killCaches(), R, SingleChannel::setClipPyramid(), ImageSet< T >::size(), SingleChannel::storePyramid(), ComplexChannel::subChan(), and uint.

**10.139.3.2 void JunctionChannel::exportOptions (const int *whatToExport*, const bool *recurse* = true)** [virtual]

Export some of our [ModelParam](#) parameters as command-line-options.

See the base function in [ModelComponent.H](#) for info.

Reimplemented from [SingleChannel](#).

Definition at line 1713 of file Channels.C.

References SingleChannel::exportOptions(), itsFull, OPTEXP\_CORE, and ModelManager::requestOption().

**10.139.4 Member Data Documentation****10.139.4.1 ModelParam<bool> JunctionChannel::itsFull** [protected]

do we want a slow, full implementation of this junction which checks for both the presence of relevant features and the absence of irrelevant features; or a quick, partial implementation which checks only for the presence of the relevant features.

Definition at line 454 of file Channels.H.

Referenced by `doInput()`, and `exportOptions()`.

#### 10.139.4.2 [VisualCortex\\*](#) [JunctionChannel::itsVCX](#) [protected]

keep track of our [VisualCortex](#)

Definition at line 457 of file `Channels.H`.

Referenced by `doInput()`.

The documentation for this class was generated from the following files:

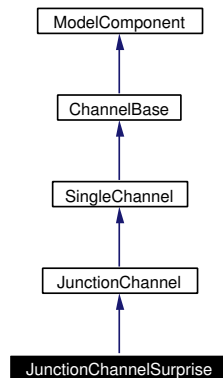
- [Channels.H](#)
- [Channels.C](#)

## 10.140 JunctionChannelSurprise Class Reference

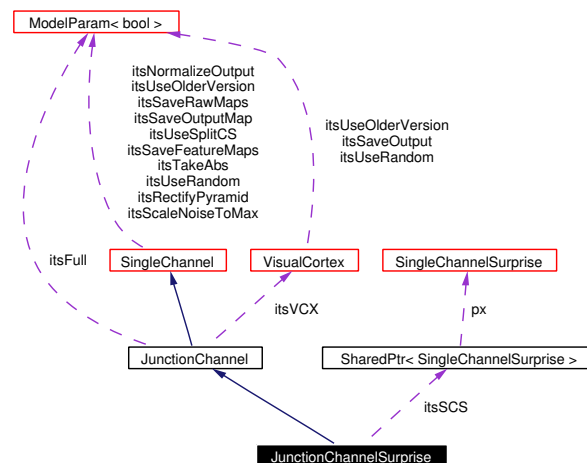
A junction channel.

```
#include <ChannelsSurprise.H>
```

Inheritance diagram for JunctionChannelSurprise:



Collaboration diagram for JunctionChannelSurprise:



### Public Member Functions

- **JunctionChannelSurprise** (**ModelManager** &mgr, **VisualCortex** \*vcx, const **VisualFeature** vs=UNKNOWN, const int R0=0, const int R1=0, const int R2=0,

const int R3=0, const int R4=0, const int R5=0, const int R6=0, const int R7=0)

*Constructor.*

- virtual [~JunctionChannelSurprise](#) ()

*Destructor.*

- virtual [Image](#)< float > [computeSubmap](#) (const [uint](#) index) const  
*[computeSubmap\(\)](#) is special and will return surprise*

## Protected Attributes

- [SharedPtr](#)< [SingleChannelSurprise](#) > [itsSCS](#)

*our surprise computer*

### 10.140.1 Detailed Description

A junction channel.

Definition at line 377 of file ChannelsSurprise.H.

### 10.140.2 Constructor & Destructor Documentation

**10.140.2.1 JunctionChannelSurprise::JunctionChannelSurprise**  
([ModelManager](#) & mgr, [VisualCortex](#) \* vcx, const [VisualFeature](#) vs = UNKNOWN, const int R0 = 0, const int R1 = 0, const int R2 = 0, const int R3 = 0, const int R4 = 0, const int R5 = 0, const int R6 = 0, const int R7 = 0)

*Constructor.*

Definition at line 343 of file ChannelsSurprise.C.

References [ModelComponent::addSubComponent\(\)](#).

**10.140.2.2 JunctionChannelSurprise::~~JunctionChannelSurprise** ()  
[virtual]

*Destructor.*

Definition at line 354 of file ChannelsSurprise.C.

### 10.140.3 Member Function Documentation

#### 10.140.3.1 [Image](#)< float > JunctionChannelSurprise::computeSubmap (const [uint](#) *index*) const [virtual]

[computeSubmap\(\)](#) is special and will return surprise

Reimplemented from [SingleChannel](#).

Definition at line 357 of file ChannelsSurprise.C.

References [uint](#).

### 10.140.4 Member Data Documentation

#### 10.140.4.1 [SharedPtr](#)<[SingleChannelSurprise](#)> JunctionChannelSurprise::its-SCS [protected]

our surprise computer

Definition at line 395 of file ChannelsSurprise.H.

The documentation for this class was generated from the following files:

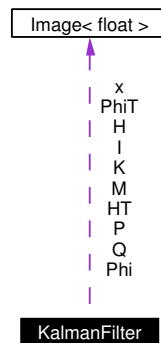
- [ChannelsSurprise.H](#)
- ChannelsSurprise.C

## 10.141 KalmanFilter Class Reference

implementation of a 2nd order linear Kalman Filter

```
#include <KalmanFilter.H>
```

Collaboration diagram for KalmanFilter:



### Public Member Functions

- [KalmanFilter](#) ()  
*default constructor - need to call init before the filter can be used*
- [KalmanFilter](#) (float initialM, float pNoise=DEF\_PNOISE, float mNoise=DEF\_MNOISE, float timeStep=DEF\_TSTEP)  
*constructor that initializes the filter*
- void [init](#) (float initialM, float pNoise=DEF\_PNOISE, float mNoise=DEF\_MNOISE, float timeStep=DEF\_TSTEP)  
*initialize the filter*
- void [writeToStream](#) (std::ostream &os) const  
*write the entire KalmanFilter to the output stream os*
- void [readFromStream](#) (std::istream &is)  
*read the KalmanFilter from the input stream is*
- float [getEstimate](#) () const  
*returns a prediction for the next value*
- float [getEstimate](#) (float measurement) const

*returns a prediction for the next value given a measurement*

- float `getSpeed ()` const  
*returns the speed (second entry in the state vector)*
- float `getCost (float measurement)` const  
*returns the cost for associating measurement with this Kalman tracker*
- float `update ()`  
*update the filter for the next time step without a measurement (skipped value)*
- float `update (float measurement)`  
*update the filter for the next time step with a measurement*
- `Image< float > getStateVector ()` const  
*returns the current state vector  $[x, v, a]$*
- `Image< float > getCovariances ()` const  
*returns the covariance matrix  $P$*
- bool `isInitialized ()` const  
*test whether the filter is initialized*

### 10.141.1 Detailed Description

implementation of a 2nd order linear Kalman Filter

This class can be used for tracking. The state vector is second order, i.e. it maintains location, speed, and acceleration to generate predictions.

Definition at line 52 of file KalmanFilter.H.

### 10.141.2 Constructor & Destructor Documentation

#### 10.141.2.1 KalmanFilter::KalmanFilter ()

default constructor - need to call init before the filter can be used

Definition at line 50 of file KalmanFilter.C.

#### 10.141.2.2 **KalmanFilter::KalmanFilter** (float *initialM*, float *pNoise* = DEF\_PNOISE, float *mNoise* = DEF\_MNOISE, float *timeStep* = DEF\_TSTEP)

constructor that initializes the filter

##### **Parameters:**

***initialM*** - the initial measurement used to jump start the filter

***pNoise*** - std of the (Gaussian) process noise

***mNoise*** - std of the (Gaussian) measurement noise

***timeStep*** - the time interval between measurements (default: 1)

Definition at line 55 of file KalmanFilter.C.

References `init()`.

### 10.141.3 Member Function Documentation

#### 10.141.3.1 **float KalmanFilter::getCost** (float *measurement*) const

returns the cost for associating measurement with this Kalman tracker

Definition at line 226 of file KalmanFilter.C.

References `ASSERT`, `getEstimate()`, and `writeToStream()`.

Referenced by `VisualEvent::getCost()`.

#### 10.141.3.2 **Image< float > KalmanFilter::getCovariances** () const

returns the covariance matrix P

Definition at line 281 of file KalmanFilter.C.

References `ASSERT`.

#### 10.141.3.3 **float KalmanFilter::getEstimate** (float *measurement*) const

returns a prediction for the next value given a measurement

Definition at line 210 of file KalmanFilter.C.

References `ASSERT`, and `Image< float >::getVal()`.



#### 10.141.3.4 float KalmanFilter::getEstimate () const

returns a prediction for the next value

Definition at line 201 of file KalmanFilter.C.

References ASSERT, and Image< float >::getVal().

Referenced by VisualEvent::assign(), VisualEvent::getCost(), getCost(), and VisualEvent::predictedLocation().

#### 10.141.3.5 float KalmanFilter::getSpeed () const

returns the speed (second entry in the state vector)

Definition at line 219 of file KalmanFilter.C.

References ASSERT, and Image< float >::getVal().

#### 10.141.3.6 Image< float > KalmanFilter::getStateVector () const

returns the current state vector [x, v, a]

Definition at line 274 of file KalmanFilter.C.

References ASSERT.

#### 10.141.3.7 void KalmanFilter::init (float *initialM*, float *pNoise* = DEF\_PNOISE, float *mNoise* = DEF\_MNOISE, float *timeStep* = DEF\_TSTEP)

initialize the filter

##### Parameters:

***initialM*** - the initial measurement used to jump start the filter

***pNoise*** - std of the (Gaussian) process noise

***mNoise*** - std of the (Gaussian) measurement noise

***timeStep*** - the time interval between measurements (default: 1)

Definition at line 64 of file KalmanFilter.C.

References eye(), NO\_INIT, Image< float >::resize(), Image< float >::setVal(), transpose(), uint, and ZEROS.

Referenced by KalmanFilter().

**10.141.3.8 bool KalmanFilter::isInitialized () const**

test whether the filter is initialized

Definition at line 288 of file KalmanFilter.C.

**10.141.3.9 void KalmanFilter::readFromStream (std::istream & is)**

read the KalmanFilter from the input stream is

Definition at line 164 of file KalmanFilter.C.

References readImageFromStream().

Referenced by VisualEvent::readFromStream().

**10.141.3.10 float KalmanFilter::update (float *measurement*)**

update the filter for the next time step with a measurement

Definition at line 258 of file KalmanFilter.C.

References ASSERT, Image< float >::getVal(), and writeToStream().

**10.141.3.11 float KalmanFilter::update ()**

update the filter for the next time step without a measurement (skipped value)

Definition at line 242 of file KalmanFilter.C.

References ASSERT, Image< float >::getVal(), and writeToStream().

Referenced by VisualEvent::assign().

**10.141.3.12 void KalmanFilter::writeToStream (std::ostream & os) const**

write the entire KalmanFilter to the output stream os

Definition at line 141 of file KalmanFilter.C.

References writeImageToStream().

Referenced by getCost(), update(), and VisualEvent::writeToStream().

The documentation for this class was generated from the following files:

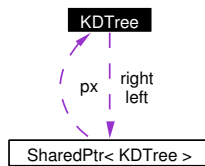
- [KalmanFilter.H](#)
- [KalmanFilter.C](#)

## 10.142 KDTree Class Reference

A simple K-d tree implementation.

```
#include <KDTree.H>
```

Collaboration diagram for KDTree:



### Public Member Functions

- [KDTree](#) ()  
*Constructor.*
- const bool [isInitialized](#) () const  
*Whether or not the tree has any values.*
- Keypoint [nearestNeighbor](#) (const Keypoint &target, double &resDist) const  
*Find the nearest neighbour to hyperspace point 'target' in kd-tree.*
- [SortedLimitedList< BestEntry > nearestNeighborList](#) (const Keypoint &target, double &resDist, int q) const  
*Find a list of the nearest-neighbors to the target in kd-tree of size q.*
- [SortedLimitedList< BestEntry > nearestNeighborListBBF](#) (const Keypoint &target, int q, int searchSteps) const  
*Limited Best-Bin-First k-d-tree nearest neighbour search.*
- bool [operator==](#) (const [KDTree](#) &tree) const  
*Find wether or not this tree is equal to the specified tree.*

### Static Public Member Functions

- [SharedPtr< KDTree > createKDTree](#) (const [std::vector](#)< Keypoint > &exset)  
*Build a kd-tree from a list of elements.*

### 10.142.1 Detailed Description

A simple K-d tree implementation.

Definition at line 306 of file KDTree.H.

### 10.142.2 Constructor & Destructor Documentation

#### 10.142.2.1 KDTree::KDTree ()

Constructor.

Definition at line 248 of file KDTree.C.

Referenced by createKDTree().

### 10.142.3 Member Function Documentation

#### 10.142.3.1 `SharedPtr< KDTree > KDTree::createKDTree (const std::vector< Keypoint > & exset) [static]`

Build a kd-tree from a list of elements.

All elements must have the same dimensionality.

Definition at line 312 of file KDTree.C.

References KDTree(), makeShared(), and uint.

Referenced by VisualObject::match().

#### 10.142.3.2 `const bool KDTree::isInitialized () const`

Whether or not the tree has any values.

Definition at line 254 of file KDTree.C.

#### 10.142.3.3 `Keypoint KDTree::nearestNeighbor (const Keypoint & target, double & resDist) const`

Find the nearest neighbour to hyperspace point 'target' in kd-tree.

After return 'resDist' contains the absolute distance from the target point. The nearest neighbour is returned.

Definition at line 260 of file KDTree.C.

References HyperRectangle::createUniverseRectangle(), and sqrt().

#### 10.142.3.4 [SortedLimitedList](#)< [BestEntry](#) > KDTree::nearestNeighborList (const Keypoint & *target*, double & *resDist*, int *q*) const

Find a list of the nearest-neighbors to the target in kd-tree of size q.

Definition at line 273 of file KDTree.C.

References [HyperRectangle::createUniverseRectangle\(\)](#), and [sqrt\(\)](#).

#### 10.142.3.5 [SortedLimitedList](#)< [BestEntry](#) > KDTree::nearestNeighborListBBF (const Keypoint & *target*, int *q*, int *searchSteps*) const

Limited Best-Bin-First k-d-tree nearest neighbour search.

Find the approximate nearest neighbour to the hyperspace point 'target' within the kd-tree using 'searchSteps' tail recursions at most (each recursion deciding about one neighbours' fitness). After return 'resDist' contains the absolute distance of the approximate nearest neighbour from the target point. The approximate nearest neighbour is returned.

Definition at line 290 of file KDTree.C.

References [HyperRectangle::createUniverseRectangle\(\)](#).

#### 10.142.3.6 bool KDTree::operator==(const [KDTree](#) & *tree*) const

Find whether or not this tree is equal to the specified tree.

Definition at line 626 of file KDTree.C.

References [dr](#), [left](#), and [right](#).

The documentation for this class was generated from the following files:

- [KDTree.H](#)
- [KDTree.C](#)

## 10.143 KeyBoard Class Reference

Simple interface to a keyboard.

```
#include <KeyBoard.H>
```

### Public Member Functions

- [KeyBoard \(\)](#)  
*Constructor.*
- [~KeyBoard \(\)](#)  
*Destructor.*
- [KeyBoardKey getKey](#) (const bool block)  
*Get a key from the keyboard.*

### 10.143.1 Detailed Description

Simple interface to a keyboard.

This class provides basic methods to check whether a key was pressed on the keyboard. It is used by the [Beobot](#), which has a small keypad with just 5 keys.

Definition at line 52 of file KeyBoard.H.

### 10.143.2 Constructor & Destructor Documentation

#### 10.143.2.1 KeyBoard::KeyBoard ()

Constructor.

Definition at line 45 of file KeyBoard.C.

#### 10.143.2.2 KeyBoard::~~KeyBoard ()

Destructor.

Definition at line 63 of file KeyBoard.C.

### 10.143.3 Member Function Documentation

#### 10.143.3.1 [KeyBoardKey](#) KeyBoard::getKey (const bool *block*)

Get a key from the keyboard.

**Parameters:**

***block*** if true, this function will block until a key is pressed. Otherwise, it will return the key pressed if any, or KBD\_NONE if no key was pressed.

Definition at line 84 of file KeyBoard.C.

References [KeyBoardKey](#).

Referenced by [main\(\)](#).

The documentation for this class was generated from the following files:

- [KeyBoard.H](#)
- [KeyBoard.C](#)

## 10.144 KeypointMatch Class Reference

KeypointMatch.

```
#include <VisualObject.H>
```

### Public Member Functions

- [KeypointMatch](#) ()  
*Constructor.*
- [KeypointMatch](#) (const Keypoint &k1, const Keypoint &k2)  
*Constructor.*
- const Keypoint & [getKeypoint1](#) () const  
*keypoint 1 getter*
- const Keypoint & [getKeypoint2](#) () const  
*keypoint 2 getter*
- const int [getDistanceSq](#) () const  
*distance squared getter*
- int [compareTo](#) (const [KeypointMatch](#) &match) const  
*compare to another match*
- bool [equals](#) (const [KeypointMatch](#) &match) const  
*return if this match equals the specified match*

### 10.144.1 Detailed Description

KeypointMatch.

Definition at line 48 of file VisualObject.H.

### 10.144.2 Constructor & Destructor Documentation

#### 10.144.2.1 KeypointMatch::KeypointMatch ()

Constructor.

Definition at line 54 of file VisualObject.C.



### 10.144.2.2 KeypointMatch::KeypointMatch (const Keypoint & *k1*, const Keypoint & *k2*)

Constructor.

Definition at line 59 of file VisualObject.C.

## 10.144.3 Member Function Documentation

### 10.144.3.1 int KeypointMatch::compareTo (const KeypointMatch & *match*) const

compare to another match

Definition at line 85 of file VisualObject.C.

References `getDistanceSq()`.

### 10.144.3.2 bool KeypointMatch::equals (const KeypointMatch & *match*) const

return if this match equals the specified match

Definition at line 96 of file VisualObject.C.

References `getKeypoint1()`, and `getKeypoint2()`.

### 10.144.3.3 const int KeypointMatch::getDistanceSq () const

distance squared getter

Definition at line 79 of file VisualObject.C.

Referenced by `VisualObjectMatch::calculateScore()`, and `compareTo()`.

### 10.144.3.4 const Keypoint & KeypointMatch::getKeypoint1 () const

keypoint 1 getter

Definition at line 67 of file VisualObject.C.

Referenced by `equals()`, and `VisualObjectMatch::writeOutput()`.

### 10.144.3.5 const Keypoint & KeypointMatch::getKeypoint2 () const

keypoint 2 getter

Definition at line 73 of file VisualObject.C.

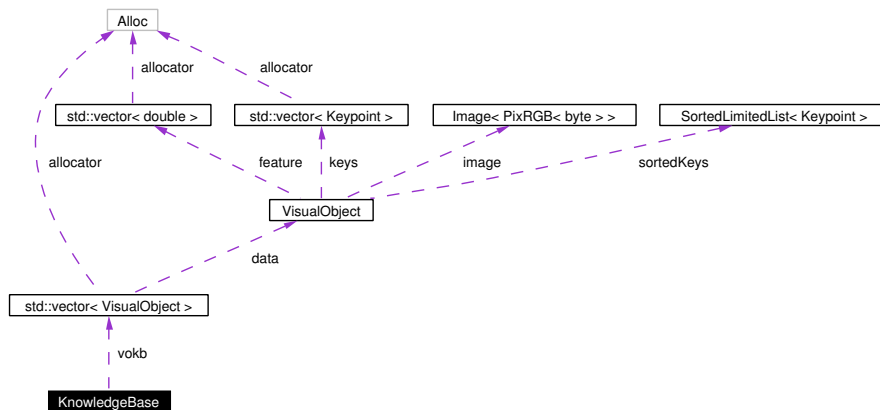
Referenced by `equals()`, and `VisualObjectMatch::writeOutput()`.

The documentation for this class was generated from the following files:

- [VisualObject.H](#)
- [VisualObject.C](#)

this class holds a knowledge base

Collaboration diagram for KnowledgeBase:



- `KnowledgeBase ()`  
*Constructor.*
- `~KnowledgeBase ()`  
*Destructor.*
- `bool load (const char *fname)`  
*load from disk*
- `bool save (const char *fname) const`  
*save to disk*
- `bool addObject (VisualObject &o)`  
*Add new object to knowledge base. True on success.*
- `bool findVisualObject (const char *name, VisualObject &o) const`  
*Find visual object from KB that has corresponding name.*
- `bool findVisualObject (const int index, VisualObject &o) const`

*Find visual object indexed by "index" in the kb array.*

- `int findVisualObjectIndex (const char *name) const`  
*Return visual object from KB that has corresponding name.*
- `VisualObject getVisualObject (const int index) const`  
*get visual object at given index:*
- `int getSize () const`  
*Return number of visual objects held in this knowledge base.*

### 10.145.1 Detailed Description

this class holds a knowledge base

Definition at line 44 of file KnowledgeBase.H.

### 10.145.2 Constructor & Destructor Documentation

#### 10.145.2.1 KnowledgeBase::KnowledgeBase ()

Constructor.

Definition at line 42 of file KnowledgeBase.C.

#### 10.145.2.2 KnowledgeBase::~KnowledgeBase ()

Destructor.

Definition at line 48 of file KnowledgeBase.C.

### 10.145.3 Member Function Documentation

#### 10.145.3.1 bool KnowledgeBase::addVisualObject (VisualObject & o)

Add new object to knowledge base. True on success.

Definition at line 83 of file KnowledgeBase.C.

References `findVisualObjectIndex()`, and `VisualObject::getName()`.

Referenced by `load()`.

**10.145.3.2** `bool KnowledgeBase::findVisualObject (const int index,  
VisualObject & o) const [inline]`

Find visual object indexed by "index" in the kb array.

returns true on success

Definition at line 96 of file KnowledgeBase.H.

References `getSize()`.

**10.145.3.3** `bool KnowledgeBase::findVisualObject (const char * name,  
VisualObject & o) const [inline]`

Find visual object from KB that has corresponding name.

returns true on success

Definition at line 87 of file KnowledgeBase.H.

References `findVisualObjectIndex()`.

**10.145.3.4** `int KnowledgeBase::findVisualObjectIndex (const char * name)  
const [inline]`

Return visual object from KB that has corresponding name.

or -1 of object not found

Definition at line 105 of file KnowledgeBase.H.

References `getSize()`.

Referenced by `addVisualObject()`, `findVisualObject()`, and `BPnnnet::train()`.

**10.145.3.5** `int KnowledgeBase::getSize () const [inline]`

Return number of visual objects held in this knowledge base.

Definition at line 121 of file KnowledgeBase.H.

Referenced by `BPnnnet::BPnnnet()`, `findVisualObject()`, `findVisualObjectIndex()`, `getVisualObject()`, and `save()`.

**10.145.3.6** `VisualObject KnowledgeBase::getVisualObject (const int index)  
const [inline]`

get visual object at given index:

Definition at line 114 of file KnowledgeBase.H.

References `getSize()`.

Referenced by `BPnnnet::recognize()`.

#### **10.145.3.7   `bool KnowledgeBase::load (const char * fname)`**

load from disk

Definition at line 52 of file `KnowledgeBase.C`.

References `addVisualObject()`.

#### **10.145.3.8   `bool KnowledgeBase::save (const char * fname) const`**

save to disk

Definition at line 72 of file `KnowledgeBase.C`.

References `getSize()`.

The documentation for this class was generated from the following files:

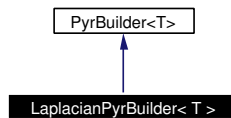
- [KnowledgeBase.H](#)
- [KnowledgeBase.C](#)

## 10.146 LaplacianPyrBuilder< T > Class Template Reference

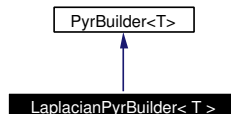
Builds pyramids based on Laplacian filters.

```
#include <PyrBuilder.H>
```

Inheritance diagram for LaplacianPyrBuilder< T >:



Collaboration diagram for LaplacianPyrBuilder< T >:



### Public Member Functions

- [LaplacianPyrBuilder](#) (const int filter\_size)  
*Constructor.*
- virtual [LaplacianPyrBuilder](#)< T > \* [clone](#) () const  
*Cloning constructor:.*
- virtual [ImageSet](#)< T > [build](#) (const [Image](#)< T > &img, const int depth)  
*Builds a Laplacian pyramid based on img.*

### 10.146.1 Detailed Description

```
template<class T> class LaplacianPyrBuilder< T >
```

Builds pyramids based on Laplacian filters.

Definition at line 167 of file PyrBuilder.H.

## 10.146.2 Constructor & Destructor Documentation

### 10.146.2.1 `template<class T> LaplacianPyrBuilder< T >::LaplacianPyrBuilder (const int filter_size)`

Constructor.

Definition at line 160 of file `PyrBuilder.C`.

## 10.146.3 Member Function Documentation

### 10.146.3.1 `template<class T> ImageSet< T > LaplacianPyrBuilder< T >::build (const Image< T > & img, const int depth)` [virtual]

Builds a Laplacian pyramid based on *img*.

Implements [PyrBuilder](#)< T >.

Definition at line 165 of file `PyrBuilder.C`.

References `buildPyrLaplacian()`.

### 10.146.3.2 `template<class T> LaplacianPyrBuilder< T > * LaplacianPyrBuilder< T >::clone () const` [inline, virtual]

Cloning constructor:.

Implements [PyrBuilder](#)< T >.

Definition at line 169 of file `PyrBuilder.C`.

The documentation for this class was generated from the following files:

- [PyrBuilder.H](#)
- [PyrBuilder.C](#)

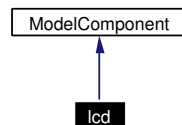


## 10.147 lcd Class Reference

[lcd.H](#) Interface to an LCD screen via serial port

```
#include <lcd.H>
```

Inheritance diagram for lcd:



Collaboration diagram for lcd:

### Public Member Functions

- [lcd](#) ([ModelManager](#) &mgr, const [std::string](#) &descrName="Alphanumeric LCD Display", const [std::string](#) &tagName="lcd")  
*Default constructor. See [ModelComponent.H](#).*
- [~lcd](#) ()  
*Destructor.*
- bool [printf](#) (const int x, const int y, const char \*fmt,...)  
*print some text starting at given cursor position*
- bool [writeString](#) (const char \*s)  
*Takes a string and outputs to LCD.*
- bool [clear](#) ()  
*Clears LCD screen.*
- bool [scrollLeft](#) (const int i)  
*Scrolls display left by i positions.*
- bool [scrollRight](#) (const int i)  
*Scrolls display right by i positions.*
- bool [home](#) ()  
*Moves cursor to start of screen.*

- bool [moveCursorLeft](#) (const int i)  
*Moves cursor left by i positions.*
- bool [moveCursorRight](#) (const int i)  
*Moves cursor right by i positions.*
- bool [blinkBlock](#) ()  
*Makes cursor a blinking block.*
- bool [blinkUnderline](#) ()  
*Makes cursor a blinking underline.*
- bool [invisibleCursor](#) ()  
*Makes cursor invisible (not recommended).*
- bool [setCursorPosition](#) (const int i)  
*Sets cursor position to i.*

### Protected Attributes

- [SharedPtr< Serial > itsPort](#)  
*our serial port*
- [ModelParam< int > itsRows](#)  
*number of rows (lines of text)*
- [ModelParam< int > itsCols](#)  
*number of columns*

### 10.147.1 Detailed Description

[lcd.H](#) Interface to an LCD screen via serial port

LCD Class. Contains interfaces to display to a LCD connected via a serial port. If you need more than a serial port to connect to an LCD, that LCD must be very special. Note that the code is made for Backpack LCDs by SEETRON.

Definition at line 52 of file [lcd.H](#).

## 10.147.2 Constructor & Destructor Documentation

### 10.147.2.1 `lcd::lcd (ModelManager & mgr, const std::string & descrName = "Alphanumeric LCD Display", const std::string & tagName = "lcd")`

Default constructor. See [ModelComponent.H](#).

Definition at line 56 of file `lcd.C`.

References `ModelComponent::addSubComponent()`.

### 10.147.2.2 `lcd::~~lcd ()`

Destructor.

Definition at line 71 of file `lcd.C`.

## 10.147.3 Member Function Documentation

### 10.147.3.1 `bool lcd::blinkBlock ()`

Makes cursor a blinking block.

Definition at line 151 of file `lcd.C`.

### 10.147.3.2 `bool lcd::blinkUnderline ()`

Makes cursor a blinking underline.

Definition at line 155 of file `lcd.C`.

### 10.147.3.3 `bool lcd::clear ()`

Clears LCD screen.

Definition at line 107 of file `lcd.C`.

### 10.147.3.4 `bool lcd::home ()`

Moves cursor to start of screen.

Definition at line 129 of file `lcd.C`.

**10.147.3.5    bool lcd::invisibleCursor ()**

Makes cursor invisible (not recommended).

Definition at line 159 of file lcd.C.

**10.147.3.6    bool lcd::moveCursorLeft (const int i)**

Moves cursor left by i positions.

Definition at line 133 of file lcd.C.

**10.147.3.7    bool lcd::moveCursorRight (const int i)**

Moves cursor right by i positions.

Definition at line 142 of file lcd.C.

**10.147.3.8    bool lcd::printf (const int x, const int y, const char \* *fmt*, ...)**

print some text starting at given cursor position

syntax is the same as [printf\(\)](#), plus the x,y coordinates. Returns true on success.

Definition at line 75 of file lcd.C.

References ModelParam< int >::getVal(), itsCols, itsRows, setCursorPosition(), and writeString().

**10.147.3.9    bool lcd::scrollLeft (const int i)**

Scrolls display left by i positions.

Definition at line 111 of file lcd.C.

**10.147.3.10    bool lcd::scrollRight (const int i)**

Scrolls display right by i positions.

Definition at line 120 of file lcd.C.

**10.147.3.11    bool lcd::setCursorPosition (const int i)**

Sets cursor position to i.

Definition at line 163 of file lcd.C.

Referenced by `printf()`.

#### 10.147.3.12 `bool lcd::writeString (const char * s)`

Takes a string and outputs to LCD.

Definition at line 103 of file `lcd.C`.

Referenced by `printf()`.

### 10.147.4 Member Data Documentation

#### 10.147.4.1 `ModelParam<int> lcd::itsCols` [protected]

number of columns

Definition at line 104 of file `lcd.H`.

Referenced by `printf()`.

#### 10.147.4.2 `SharedPtr<Serial> lcd::itsPort` [protected]

our serial port

Definition at line 102 of file `lcd.H`.

#### 10.147.4.3 `ModelParam<int> lcd::itsRows` [protected]

number of rows (lines of text)

Definition at line 103 of file `lcd.H`.

Referenced by `printf()`.

The documentation for this class was generated from the following files:

- [lcd.H](#)
- [lcd.C](#)

## 10.148 LeakyIntegrator Class Reference

A leaky integrator neuron, used by standard saliency map (SM).

```
#include <LeakyIntegrator.H>
```

### Public Member Functions

- [LeakyIntegrator](#) (const double timeStep=0.0001, const float C=5.0E-8F, const float Gleak=1.0E-7F, const float GinhDecay=0.9999F)  
*constructor*
- void [input](#) (const float current)  
*set input current (A)*
- void [resetV](#) (const double val)  
*reset membrane potential to given value (in Volts)*
- void [integrate](#) (const double t)  
*integrate for up to given time (in s)*
- float [getV](#) () const  
*get current membrane potential (in V)*
- void [setGinh](#) (const float g)  
*set inhibitory conductance (S)*
- void [addGinh](#) (const float g)  
*add to inhibitory conductance (S)*
- void [setGinhDecay](#) (const float decay)  
*set the decay factor for Ginh*

### 10.148.1 Detailed Description

A leaky integrator neuron, used by standard saliency map (SM).

This is the saliency map neuron used in conjunction with SM which is a 2D array of LeakyIntegrator.

Definition at line 45 of file LeakyIntegrator.H.

## 10.148.2 Constructor & Destructor Documentation

**10.148.2.1** `LeakyIntegrator::LeakyIntegrator (const double timeStep = 0.0001, const float C = 5.0E-8F, const float Gleak = 1.0E-7F, const float GinhDecay = 0.9999F) [inline]`

constructor

Constructor with default params

**Parameters:**

*timeStep* is the integration time step, in s

*C* is the membrane capacitance, in Farads

*Gleak* is the leak conductance, in Siemens

*GinhDecay* decay factor applied to *Ginh* at every time step

Definition at line 95 of file LeakyIntegrator.H.

## 10.148.3 Member Function Documentation

**10.148.3.1** `void LeakyIntegrator::addGinh (const float g) [inline]`

add to inhibitory conductance (S)

Definition at line 143 of file LeakyIntegrator.H.

**10.148.3.2** `float LeakyIntegrator::getV () const [inline]`

get current membrane potential (in V)

Definition at line 135 of file LeakyIntegrator.H.

Referenced by SaliencyMapStd::getV().

**10.148.3.3** `void LeakyIntegrator::input (const float current) [inline]`

set input current (A)

Definition at line 104 of file LeakyIntegrator.H.

**10.148.3.4** `void LeakyIntegrator::integrate (const double t) [inline]`

integrate for up to given time (in s)

Definition at line 112 of file LeakyIntegrator.H.

**10.148.3.5 void LeakyIntegrator::resetV (const double *val*) [inline]**

reset membrane potential to given value (in Volts)

Definition at line 108 of file LeakyIntegrator.H.

**10.148.3.6 void LeakyIntegrator::setGinh (const float *g*) [inline]**

set inhibitory conductance (S)

Definition at line 139 of file LeakyIntegrator.H.

**10.148.3.7 void LeakyIntegrator::setGinhDecay (const float *decay*) [inline]**

set the decay factor for Ginh

Definition at line 147 of file LeakyIntegrator.H.

The documentation for this class was generated from the following file:

- [LeakyIntegrator.H](#)



## 10.149 LeakyIntFire Class Reference

A leaky integrate & fire neuron, used by standard winner-take-all (WTA).

```
#include <LeakyIntFire.H>
```

### Public Member Functions

- [LeakyIntFire](#) (const double timeStep=0.0001, const float El=0.0F, const float Ee=100.0e-3F, const float Ei=-20.0e-3F, const float Gl=5.0e-8F, const float Ge=0.0F, const float Gi=0.0F, const float Vth=0.001F, const float C=1.0E-9F)  
*constructor*
- void [input](#) (const float current)  
*set input current (A)*
- void [setV](#) (const double val)  
*set membrane potential to given value relative to Ei (in Volts)*
- bool [integrate](#) (const double t)  
*integrate for up to given time (in s)*
- float [getV](#) () const  
*get current membrane potential (in V)*
- void [setG](#) (const float Exc, const float Inh)  
*set excitatory and inhibitory conductances (S)*
- void [setGleak](#) (const float Leak)  
*set leak conductance (S)*
- double [getTimeStep](#) () const  
*return our internal time step:*

### 10.149.1 Detailed Description

A leaky integrate & fire neuron, used by standard winner-take-all (WTA).

This is the integrate & fire neuron used in conjunction with WTA which is a 2D array of LeakyIntFire. It has a slightly more complicated Euler equation then [LeakyIntegrator](#) used by SM but the basic idea is the same. All parameters are in MKSA (SI) units (i.e.,

volts, amperes, siemens, etc). This neuron uses an Euler integration of the following form:

At each time step,

$$V \leftarrow V + \frac{\delta t}{C} (I - G_{leak}(V - E_{leak}) - G_{ex}(V - E_{ex}) - G_{inh}(V - E_{inh}))$$

V is also clamped to never become smaller than  $E_{inh}$  and a spike is generated when  $V \geq V_{th}$  (and V is then reset to zero).

Definition at line 57 of file LeakyIntFire.H.

## 10.149.2 Constructor & Destructor Documentation

**10.149.2.1 LeakyIntFire::LeakyIntFire (const double *timeStep* = 0.0001, const float *El* = 0.0F, const float *Ee* = 100.0e-3F, const float *Ei* = -20.0e-3F, const float *Gl* = 5.0e-8F, const float *Ge* = 0.0F, const float *Gi* = 0.0F, const float *Vth* = 0.001F, const float *C* = 1.0E-9F) [inline]**

constructor

Constructor with default params

### Parameters:

***timeStep*** is the integration time step, in s. See the general description for the other params.

Definition at line 117 of file LeakyIntFire.H.

## 10.149.3 Member Function Documentation

**10.149.3.1 double LeakyIntFire::getTimeStep () const [inline]**

return our internal time step:

Definition at line 187 of file LeakyIntFire.H.

Referenced by WinnerTakeAllStd::evolve().

**10.149.3.2 float LeakyIntFire::getV () const [inline]**

get current membrane potential (in V)

Definition at line 171 of file LeakyIntFire.H.

Referenced by WinnerTakeAllStd::getV().

**10.149.3.3 void LeakyIntFire::input (const float *current*) [inline]**

set input current (A)

Definition at line 131 of file LeakyIntFire.H.

**10.149.3.4 bool LeakyIntFire::integrate (const double *t*) [inline]**

integrate for up to given time (in s)

Returns true if a spike was generated.

Definition at line 135 of file LeakyIntFire.H.

Referenced by WinnerTakeAllStd::evolve().

**10.149.3.5 void LeakyIntFire::setG (const float *Exc*, const float *Inh*) [inline]**

set excitatory and inhibitory conductances (S)

Definition at line 175 of file LeakyIntFire.H.

Referenced by WinnerTakeAllStd::evolve().

**10.149.3.6 void LeakyIntFire::setGleak (const float *Leak*) [inline]**

set leak conductance (S)

Definition at line 179 of file LeakyIntFire.H.

Referenced by WinnerTakeAllStd::WinnerTakeAllStd().

**10.149.3.7 void LeakyIntFire::setV (const double *val*) [inline]**

set membrane potential to given value relative to Ei (in Volts)

Definition at line 167 of file LeakyIntFire.H.

The documentation for this class was generated from the following file:

- [LeakyIntFire.H](#)

## 10.150 LevelSpec Class Reference

A utility class for use with [SingleChannel](#).

```
#include <LevelSpec.H>
```

### Public Member Functions

- [LevelSpec](#) ()  
*Unitialized constructor, all internal values to 0.*
- [LevelSpec](#) (const [uint](#) levmin, const [uint](#) levmax, const [uint](#) deltmin, const [uint](#) deltmax, const [uint](#) maplevel)  
*Constructor.*
- void [init](#) (const [uint](#) levmin, const [uint](#) levmax, const [uint](#) deltmin, const [uint](#) deltmax, const [uint](#) maplevel)  
*(re-)initialization*
- [uint levMin](#) () const  
*Get the minimum center level (inclusive).*
- [uint levMax](#) () const  
*Get the maximum center level (inclusive).*
- [uint delMin](#) () const  
*Get the minimum surround minus center level difference (inclusive).*
- [uint delMax](#) () const  
*Get the maximum surround minus center level difference (inclusive).*
- [uint mapLevel](#) () const  
*Get the level at which the feature map is built.*
- [uint maxIndex](#) () const  
*Get the maximum index that can be returned from [csToIndex\(\)](#).*
- [uint maxDepth](#) () const  
*Get the maximum surround level that will be used.*
- bool [indexOK](#) ([uint](#) index) const  
*Query if the given array index is valid for this object.*

- bool [cleverOK](#) (uint centerlev) const  
*Query if the given center level is valid for this object.*
- bool [sleverOK](#) (uint surroundlev) const  
*Query if the given surround level is valid for this object.*
- bool [delOK](#) (uint delta) const  
*Query if the given surround-center delta is valid for this object.*
- bool [csOK](#) (uint centerlev, uint surroundlev) const  
*Query if the given center/surround levels are valid for this object.*
- uint [csToIndex](#) (uint centerlev, uint surroundlev) const  
*Compute an array index for a given center/surround pairing.*
- void [indexToCS](#) (uint index, uint &centerlev, uint &surroundlev) const  
*Compute the center/surround pairing corresponding to an array index.*

### 10.150.1 Detailed Description

A utility class for use with [SingleChannel](#).

LevelSpec specifies a set of center/surround pairings, by tracking two min/max ranges: one for *levels*, and one for *deltas*. *Note that both of these ranges are inclusive ranges; both the min and max values are valid elements. The levels refer to depth levels in a [centerSurround\(\)](#) operation. LevelSpec also maps these ranges into a single range [0..maxIndex\(\)](#), so that all pairings of level/delta can be accessed in a 1-D array, for example.*

Definition at line 52 of file LevelSpec.H.

### 10.150.2 Constructor & Destructor Documentation

#### 10.150.2.1 LevelSpec::LevelSpec ()

Unitialized constructor, all internal values to 0.

Definition at line 49 of file LevelSpec.C.

### 10.150.2.2 **LevelSpec::LevelSpec** (const [uint](#) *levmin*, const [uint](#) *levmax*, const [uint](#) *deltmin*, const [uint](#) *deltmax*, const [uint](#) *maplevel*)

Constructor.

Definition at line 55 of file LevelSpec.C.

References ASSERT, and uint.

## 10.150.3 Member Function Documentation

### 10.150.3.1 **bool LevelSpec::clevOK** ([uint](#) *centerlev*) const

Query if the given center level is valid for this object.

Definition at line 115 of file LevelSpec.C.

References uint.

Referenced by csOK().

### 10.150.3.2 **bool LevelSpec::csOK** ([uint](#) *centerlev*, [uint](#) *surroundlev*) const

Query if the given center/surround levels are valid for this object.

The "cs" prefix is a mnemonic to indicate that the center argument precedes the surround argument.

Definition at line 130 of file LevelSpec.C.

References clevOK(), delOK(), sleveOK(), and uint.

Referenced by SingleChannel::centerSurround(), DummyChannel::centerSurround(), csToIndex(), and indexToCS().

### 10.150.3.3 **uint LevelSpec::csToIndex** ([uint](#) *centerlev*, [uint](#) *surroundlev*) const

Compute an array index for a given center/surround pairing.

Return the array index in 0..maxIndex that corresponds to the given center/surround levels. The "cs" prefix is a mnemonic to indicate that the center argument precedes the surround argument.

Definition at line 143 of file LevelSpec.C.

References ASSERT, csOK(), and uint.

Referenced by SingleChannel::csToIndex().

#### 10.150.3.4 `uint LevelSpec::delMax () const`

Get the maximum surround minus center level difference (inclusive).

Definition at line 95 of file LevelSpec.C.

References `uint`.

Referenced by `convertToString()`, `PN03contrastChannel::doInput()`, `InformationChannel::doInput()`, `ScorrChannel::doInput()`, `TcorrChannel::doInput()`, `VarianceChannel::doInput()`, `EntropyChannel::doInput()`, `SingleChannel::fillJet()`, `SingleChannel::fillJetNI()`, `SingleChannel::fillJetSpec()`, `SingleChannel::findLikelihood()`, `SingleChannel::learn()`, `SingleChannel::readFrom()`, `SingleChannel::reco()`, and `SingleChannel::writeTo()`.

#### 10.150.3.5 `uint LevelSpec::delMin () const`

Get the minimum surround minus center level difference (inclusive).

Definition at line 91 of file LevelSpec.C.

References `uint`.

Referenced by `convertToString()`, `PN03contrastChannel::doInput()`, `InformationChannel::doInput()`, `ScorrChannel::doInput()`, `TcorrChannel::doInput()`, `VarianceChannel::doInput()`, `EntropyChannel::doInput()`, `SingleChannel::findLikelihood()`, `SingleChannel::learn()`, `SingleChannel::readFrom()`, `SingleChannel::reco()`, and `SingleChannel::writeTo()`.

#### 10.150.3.6 `bool LevelSpec::delOK (uint delta) const`

Query if the given surround-center delta is valid for this object.

Definition at line 126 of file LevelSpec.C.

References `uint`.

Referenced by `csOK()`.

#### 10.150.3.7 `bool LevelSpec::indexOK (uint index) const`

Query if the given array index is valid for this object.

Definition at line 111 of file LevelSpec.C.

References `uint`.

Referenced by `SingleChannel::biasFeatures()`, `SingleChannel::getRawCSmap()`, `SingleChannel::getSubmapName()`, and `indexToCS()`.

#### 10.150.3.8 **void LevelSpec::indexToCS (uint index, uint & centerlev, uint & surroundlev) const**

Compute the center/surround pairing corresponding to an array index.

Definition at line 154 of file LevelSpec.C.

References ASSERT, csOK(), indexOK(), and uint.

Referenced by SingleChannel::biasFeatures(), SingleChannel::combineSubMaps(), SingleChannel::findWeight(), SingleChannel::getRawCSmap(), SingleChannel::getSubmapName(), SingleChannel::postProcessMap(), and SingleChannel::saveResults().

#### 10.150.3.9 **void LevelSpec::init (const uint levmin, const uint levmax, const uint deltmin, const uint deltmax, const uint maplevel)**

(re-)initialization

Definition at line 70 of file LevelSpec.C.

References ASSERT, and uint.

Referenced by convertFromString().

#### 10.150.3.10 **uint LevelSpec::levMax () const**

Get the maximum center level (inclusive).

Definition at line 87 of file LevelSpec.C.

References uint.

Referenced by convertToString(), PN03contrastChannel::doInput(), InformationChannel::doInput(), ScorrChannel::doInput(), TcorrChannel::doInput(), VarianceChannel::doInput(), EntropyChannel::doInput(), SingleChannel::fillJet(), SingleChannel::fillJetNI(), SingleChannel::fillJetSpec(), TemplateMatchChannel::findBestMatch(), SingleChannel::findLikelihood(), SingleChannel::learn(), SingleChannel::readFrom(), SingleChannel::reco(), and SingleChannel::writeTo().

#### 10.150.3.11 **uint LevelSpec::levMin () const**

Get the minimum center level (inclusive).

Definition at line 83 of file LevelSpec.C.

References uint.

Referenced by convertToString(), PN03contrastChannel::doInput(), InformationChannel::doInput(), ScorrChannel::doInput(), TcorrChannel::doInput(), Variance-



Channel::doInput(), EntropyChannel::doInput(), TemplateMatchChannel::findBestMatch(), SingleChannel::findLikelihood(), SingleChannel::learn(), SingleChannel::readFrom(), SingleChannel::reco(), and SingleChannel::writeTo().

### 10.150.3.12 [uint](#) LevelSpec::mapLevel () const

Get the level at which the feature map is built.

Definition at line 99 of file LevelSpec.C.

References [uint](#).

Referenced by VisualBuffer::bufferToRetinal(), HueChannel::combineSubMaps(), convertToString(), PN03contrastChannel::doInput(), InformationChannel::doInput(), ScorrChannel::doInput(), TcorrChannel::doInput(), VarianceChannel::doInput(), EntropyChannel::doInput(), Brain::evolve(), SimulationViewerEyeMvt2::eyeMovement(), SimulationViewerEyeMvt::eyeMovement(), SingleChannel::getMapDims(), VisualCortex::getOutput(), TcorrChannel::getOutput(), FlickerChannel::getOutput(), SimulationViewerEyeMvt2::getTraj(), SimulationViewerCompress::getTraj(), VisualBuffer::inhibit(), VisualBuffer::retinalToBuffer(), VisualCortexEyeMvt::start1(), and TaskRelevanceMapKillStatic::start1().

### 10.150.3.13 [uint](#) LevelSpec::maxDepth () const

Get the maximum surround level that will be used.

Definition at line 107 of file LevelSpec.C.

References [uint](#).

Referenced by SingleChannel::computePyramid(), RGBConvolveChannel::computePyramid(), SoxChannel::numScales(), SingleChannel::setClipPyramid(), StereoChannel::start1(), and DisparityChannel::start1().

### 10.150.3.14 [uint](#) LevelSpec::maxIndex () const

Get the maximum index that can be returned from [csToIndex\(\)](#).

Definition at line 103 of file LevelSpec.C.

References [uint](#).

Referenced by SingleChannel::absSumCoeffs(), SingleChannel::clampCoeffs(), SingleChannel::cleanCoeffs(), SingleChannel::combineSubMaps(), SingleChannel::killCaches(), SingleChannel::normalizeCoeffs(), SingleChannel::numSubmaps(), SingleChannel::reset(), SingleChannelSurprise::start1(), and SingleChannel::start1().

**10.150.3.15** `bool LevelSpec::slevOK (uint surroundlev) const`

Query if the given surround level is valid for this object.

Definition at line 119 of file LevelSpec.C.

References `uint`.

Referenced by `csOK()`.

The documentation for this class was generated from the following files:

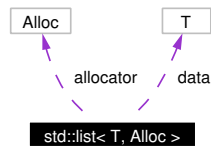
- [LevelSpec.H](#)
- [LevelSpec.C](#)

## 10.151 `std::list< T, Alloc >` Class Template Reference

The STL list container class. Use with `#include` .

`#include <dummySTL.H>`

Collaboration diagram for `std::list< T, Alloc >`:



### Public Attributes

- `T data`
- `Alloc allocator`

#### 10.151.1 Detailed Description

**template<class T, class Alloc> class `std::list< T, Alloc >`**

The STL list container class. Use with `#include` .

See [SGI's documentation](#) of `std::list`.

Definition at line 25 of file `dummySTL.H`.

The documentation for this class was generated from the following file:

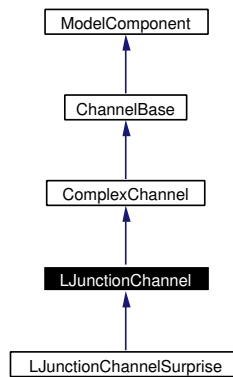
- [dummySTL.H](#)

## 10.152 LJunctionChannel Class Reference

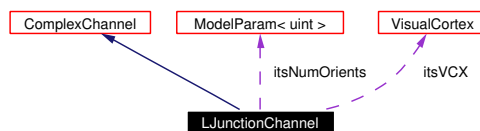
An L junction channel that detects L junctions at different.

```
#include <Channels.H>
```

Inheritance diagram for LJunctionChannel:



Collaboration diagram for LJunctionChannel:



### Public Member Functions

- `LJunctionChannel` (`ModelManager` &mgr, `VisualCortex` \*vcx)  
*Construct with standard params.*
- virtual `~LJunctionChannel` ()  
*Destructor.*
- virtual void `exportOptions` (const int whatToExport, const bool recurse=true)  
*Export some of our `ModelParam` parameters as command-line-options.*
- virtual bool `setModelParamString` (const `std::string` &name, const `std::string` &value, const bool recurse=false, const bool warn=true)

*Overload so that we can reconfigure when our params get changed.*

## Protected Member Functions

- virtual void [buildSubChans](#) ()  
*(re-)build our subchannels*
- virtual void [doInput](#) (const [Image](#)< [PixRGB](#)< float > > \*, const [Image](#)< float > \*bwimg, const double t, const [Image](#)< byte > &clipMask)  
*Computes the L junctions at different orientations.*

## Protected Attributes

- [ModelParam](#)< uint > [itsNumOrients](#)  
*number of junction subchannels*
- int [itsExportOpts](#)  
*keep track of what options to export in case we rebuild our subchans*
- [VisualCortex](#) \* [itsVCX](#)

### 10.152.1 Detailed Description

An L junction channel that detects L junctions at different.

Definition at line 1282 of file Channels.H.

### 10.152.2 Constructor & Destructor Documentation

#### 10.152.2.1 LJunctionChannel::LJunctionChannel ([ModelManager](#) & mgr, [VisualCortex](#) \* vcx)

Construct with standard params.

Definition at line 1774 of file Channels.C.

References [buildSubChans\(\)](#), [LJUNCTION](#), and [OPTEXP\\_NONE](#).

**10.152.2.2 LJunctionChannel::~~LJunctionChannel () [virtual]**

Destructor.

Definition at line 1786 of file Channels.C.

**10.152.3 Member Function Documentation****10.152.3.1 void LJunctionChannel::buildSubChans () [protected, virtual]**

(re-)build our subchannels

Reimplemented in [LJunctionChannelSurprise](#).

Definition at line 1789 of file Channels.C.

References [ComplexChannel::addSubChan\(\)](#), [ModelParam< uint >::getVal\(\)](#), [makeShared\(\)](#), [ModelComponent::removeAllSubComponents\(\)](#), [ChannelBase::setHomogeneous\(\)](#), [uint](#), and [ChannelBase::visualFeature\(\)](#).

Referenced by [LJunctionChannel\(\)](#), and [setModelParamString\(\)](#).

**10.152.3.2 void LJunctionChannel::doInput (const [Image< PixRGB< float > > \\*](#), const [Image< float > \\*](#) *bwimg*, const double *t*, const [Image< byte > & clipMask](#)) [protected, virtual]**

Computes the L junctions at different orientations.

Implements [ChannelBase](#).

Definition at line 1881 of file Channels.C.

References [ASSERT](#), [VisualCortex::hasChannel\(\)](#), [ChannelBase::input\(\)](#), [ComplexChannel::numChans\(\)](#), [ComplexChannel::subChan\(\)](#), and [uint](#).

**10.152.3.3 void LJunctionChannel::exportOptions (const int *whatToExport*, const bool *recurse* = true) [virtual]**

Export some of our [ModelParam](#) parameters as command-line-options.

See the base function in [ModelComponent.H](#) for info.

Reimplemented from [ComplexChannel](#).

Definition at line 1866 of file Channels.C.

References [ComplexChannel::exportOptions\(\)](#), [OPTEXP\\_CORE](#), and [ModelManager::requestOption\(\)](#).

**10.152.3.4** `bool LJunctionChannel::setModelParamString (const std::string & name, const std::string & value, const bool recurse = false, const bool warn = true) [virtual]`

Overload so that we can reconfigure when our params get changed.

Reimplemented from [ModelComponent](#).

Definition at line 1848 of file Channels.C.

References [buildSubChans\(\)](#), [ModelParamBase::getName\(\)](#), [ModelParam< uint >::getVal\(\)](#), [ComplexChannel::numChans\(\)](#), and [ModelComponent::setModelParamString\(\)](#).

## 10.152.4 Member Data Documentation

**10.152.4.1** `int LJunctionChannel::itsExportOpts [protected]`

keep track of what options to export in case we rebuild our subchans

Definition at line 1314 of file Channels.H.

**10.152.4.2** `ModelParam<uint> LJunctionChannel::itsNumOrients [protected]`

number of junction subchannels

Definition at line 1311 of file Channels.H.

The documentation for this class was generated from the following files:

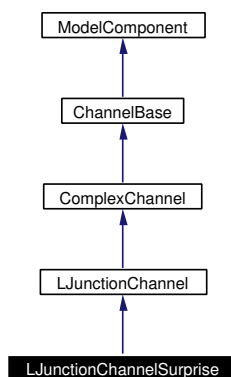
- [Channels.H](#)
- [Channels.C](#)

## 10.153 LJunctionChannelSurprise Class Reference

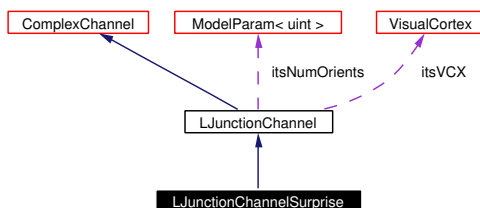
An L junction channel that detects L junctions at different orientations.

```
#include <ChannelsSurprise.H>
```

Inheritance diagram for LJunctionChannelSurprise:



Collaboration diagram for LJunctionChannelSurprise:



### Public Member Functions

- **LJunctionChannelSurprise** (**ModelManager** &mgr, **VisualCortex** \*vcx)

*Construct with standard params.*

- virtual **~LJunctionChannelSurprise** ()

*Destructor.*



## Protected Member Functions

- virtual void [buildSubChans](#) ()  
(re-)build our subchannels

### 10.153.1 Detailed Description

An L junction channel that detects L junctions at different orientations.

Definition at line 564 of file ChannelsSurprise.H.

### 10.153.2 Constructor & Destructor Documentation

#### 10.153.2.1 LJunctionChannelSurprise::LJunctionChannelSurprise ([ModelManager](#) & mgr, [VisualCortex](#) \* vcx)

Construct with standard params.

Definition at line 677 of file ChannelsSurprise.C.

References [buildSubChans\(\)](#).

#### 10.153.2.2 LJunctionChannelSurprise::~~LJunctionChannelSurprise () [virtual]

Destructor.

Definition at line 685 of file ChannelsSurprise.C.

### 10.153.3 Member Function Documentation

#### 10.153.3.1 void LJunctionChannelSurprise::buildSubChans () [protected, virtual]

(re-)build our subchannels

Reimplemented from [LJunctionChannel](#).

Definition at line 689 of file ChannelsSurprise.C.

References [ComplexChannel::addSubChan\(\)](#), [ModelParam< uint >::getVal\(\)](#), [makeShared\(\)](#), [ModelComponent::removeAllSubComponents\(\)](#), [ChannelBase::setHomogeneous\(\)](#), [uint](#), and [ChannelBase::visualFeature\(\)](#).

Referenced by [LJunctionChannelSurprise\(\)](#).

The documentation for this class was generated from the following files:

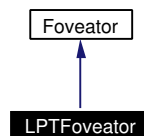
- [ChannelsSurprise.H](#)
- ChannelsSurprise.C

## 10.154 LPTFoveator Class Reference

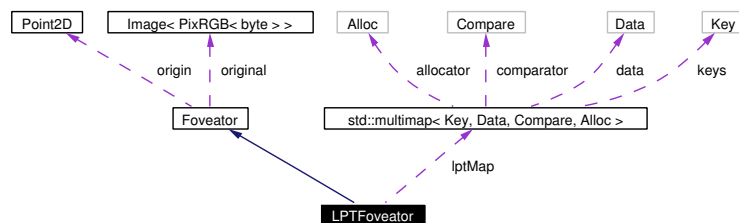
A class that does space-variant processing with log polar transform.

```
#include <LPTFoveator.H>
```

Inheritance diagram for LPTFoveator:



Collaboration diagram for LPTFoveator:



### Public Member Functions

- **LPTFoveator** (const **Image**< **PixRGB**< **byte** > > &img, int lptW, int lptH)  
*Construct a LPTFoveator for img.*
- **~LPTFoveator** ()  
*Destructor.*
- virtual bool **setOrigin** (**Point2D** pt)  
*Set origin point (center of foveation).*
- virtual bool **setOrigin** (int x, int y)  
*Set origin point (center of foveation).*
- **Image**< **PixRGB**< **byte** > > **foveate** (void)  
*Perform log-polar foveation and return resulting image.*

- `Image< PixRGB< byte > > getLPT` (void)  
*Perform log-polar transform and return map image.*
- `Image< PixRGB< byte > > invLPT` (const `Image< PixRGB< byte > > &img`)  
*Perform inverse log-polar transform on a map image to get foveated image.*

## Static Public Member Functions

- `Image< PixRGB< byte > > foveate` (const `Image< PixRGB< byte > > &img`, int `lptW`, int `lptH`, int `x`, int `y`, bool `getMap=false`)  
*Perform one log-polar foveation without a LPTFoveator.*
- `Image< PixRGB< byte > > invLPT` (const `Image< PixRGB< byte > > &img`, int `w`, int `h`, int `x`, int `y`)  
*Get foveated image from log-polar map image.*

### 10.154.1 Detailed Description

A class that does space-variant processing with log polar transform.

A log-polar transform (LPT) maps pixels of the original image to a smaller "map" image that holds pixel data. The LPT transforms rectangular  $(x, y)$  coordinates into  $(r, \theta)$ , where  $r$  is a function of the log of the radius from the point to the origin, and  $\theta$  is the angle from the horizontal axis.

For points close to the origin, the result is that the LPT mapping is almost one-to-one, while in the periphery many pixels may correspond to a single ordered pair of  $(r, \theta)$ , resulting in some degree of data loss. To display the foveated image an inverse LPT must be performed with the `invLPT()` set of methods.

LPTFoveator is very fast compared with `BlurFoveator` and `PyrFoveator`, and is useful for compression. However, there is a significant amount of blockiness in the periphery in the foveated image, and it seems that noise can have a serious effect on image quality.

Definition at line 30 of file LPTFoveator.H.

### 10.154.2 Constructor & Destructor Documentation

#### 10.154.2.1 LPTFoveator::LPTFoveator (const `Image< PixRGB< byte > > &img`, int `lptW`, int `lptH`)

Construct a LPTFoveator for `img`.

This constructor will create a LPTFoveator that handles *img* and all other images of the same dimensions.

**Parameters:**

*img* image to be foveated

*lptW* width of log-polar map image, controls radial detail

*lptH* height of log-polar map image, controls angular detail

Definition at line 9 of file LPTFoveator.C.

References `setOrigin()`.

### 10.154.2.2 LPTFoveator::~~LPTFoveator ()

Destructor.

Definition at line 21 of file LPTFoveator.C.

## 10.154.3 Member Function Documentation

### 10.154.3.1 Image< PixRGB< byte > > LPTFoveator::foveate (const Image< PixRGB< byte > > & *img*, int *lptW*, int *lptH*, int *x*, int *y*, bool *getMap* = false) [static]

Perform one log-polar foveation without a LPTFoveator.

This function is for foveating ONE image with a call such as `LPTFoveator::foveate()`. It is not optimized for foveating multiple images.

**Parameters:**

*img* image to be foveated

*lptW* width of log-polar map image, controls radial detail

*lptH* height of log-polar map image, controls angular detail

*x* horizontal coordinate of origin

*y* vertical coordinate of origin

*getMap* set this to true if you want to have this function return map image rather than foveated image

Definition at line 128 of file LPTFoveator.C.

References `Image< T >::getWidth()`, `invLPT()`, `NO_INIT`, `pix`, and `Image< T >::setVal()`.

### 10.154.3.2 `Image< PixRGB< byte > > LPTFoveator::foveate (void)` [virtual]

Perform log-polar foveation and return resulting image.

Implements [Foveator](#).

Definition at line 71 of file LPTFoveator.C.

References [getLPT\(\)](#), and [invLPT\(\)](#).

Referenced by [main\(\)](#).

### 10.154.3.3 `Image< PixRGB< byte > > LPTFoveator::getLPT (void)`

Perform log-polar transform and return map image.

Definition at line 79 of file LPTFoveator.C.

References [PixRGB< byte >::blue\(\)](#), [Image< PixRGB< byte > >::getVal\(\)](#), [PixRGB< byte >::green\(\)](#), [NO\\_INIT](#), [pix](#), [PixRGB< byte >::red\(\)](#), and [Image< T >::setVal\(\)](#).

Referenced by [foveate\(\)](#), and [main\(\)](#).

### 10.154.3.4 `Image< PixRGB< byte > > LPTFoveator::invLPT (const Image< PixRGB< byte > > & img, int w, int h, int x, int y) [static]`

Get foveated image from log-polar map image.

This function can be used to perform an inverse log-polar transform without a LPT-Foveator. It is not optimized for performing this operation on multiple images, but it would be faster to use this function on an map image than to call [LPTFoveator::foveate\(\)](#) again with `getMap = true`.

#### Parameters:

- img* image to be foveated
- w* width of original image (before LPT)
- h* height of original image (before LPT)
- x* horizontal coordinate of origin
- y* vertical coordinate of origin

Definition at line 237 of file LPTFoveator.C.

References [NO\\_INIT](#), and [Image< T >::setVal\(\)](#).

### 10.154.3.5 `Image< PixRGB< byte > > LPTFoveator::invLPT (const Image< PixRGB< byte > > &img)`

Perform inverse log-polar transform on a map image to get foveated image.

This function can do an inverse log-polar transform on an image returned by [LPTFoveator::getLPT\(\)](#). It will be important that this function must be called from the same LPTFoveator object that created the map image, or from one with the same configuration parameters. Also, the origin must not have been changed since the map image was generated.

#### Parameters:

*img* map image (source from which foveated image will be created)

Definition at line 116 of file LPTFoveator.C.

References `NO_INIT`, and `Image< T >::setVal()`.

Referenced by `foveate()`.

### 10.154.3.6 `bool LPTFoveator::setOrigin (int x, int y) [virtual]`

Set origin point (center of foveation).

Set origin point to two int arguments, returns true if successful

#### Parameters:

*x* horizontal coordinate of origin

*y* vertical coordinate of origin

Reimplemented from [Foveator](#).

Definition at line 63 of file LPTFoveator.C.

References `setOrigin()`.

### 10.154.3.7 `bool LPTFoveator::setOrigin (Point2D pt) [virtual]`

Set origin point (center of foveation).

Set origin point to [Point2D](#) argument, returns true if successful

#### Parameters:

*pt* center of foveation

Reimplemented from [Foveator](#).

Definition at line 30 of file LPTFoveator.C.

References `Point2D::i`, `Point2D::j`, and `Foveator::setOrigin()`.

Referenced by `LPTFoveator()`, and `setOrigin()`.

The documentation for this class was generated from the following files:

- [LPTFoveator.H](#)
- [LPTFoveator.C](#)

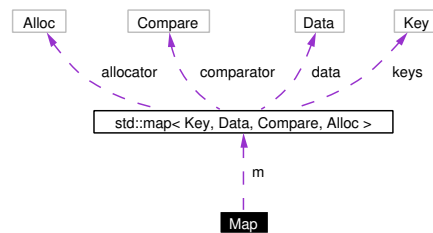


## 10.155 Map Class Reference

Definition of Map class and access functions.

```
#include <Make_map.H>
```

Collaboration diagram for Map:



### Public Member Functions

- [Map \(\)](#)  
*Constructor.*
- [~Map \(\)](#)  
*Destructor.*
- void [insert\\_element](#) ([std::string](#) name, [Entity](#) &entity)  
*Insert an element into the map.*
- const [Entity](#) \* [get](#) ([std::string](#) name) const  
*Find the element given the key string; returns a pointer to the entity.*
- void [delete\\_element](#) ([std::string](#) name)  
*Delete an element given the key string.*
- void [create\\_Map](#) (char \*file, char c, [Map](#) \*object, [Map](#) \*subject, [Map](#) \*action)  
*Reads from the XML file and makes the map.*
- void [print\\_map](#) (char \*file) const  
*Print contents of map.*
- map\_ontology::iterator [find](#) ([std::string](#) name)
- map\_ontology::iterator [end](#) ()
- void [insert](#) ([std::string](#) name, [Entity](#) entity)

### 10.155.1 Detailed Description

Definition of Map class and access functions.

Map class contains a map\_ontology that is a map of names to their corresponding Entities.

Definition at line 55 of file Make\_map.H.

### 10.155.2 Constructor & Destructor Documentation

#### 10.155.2.1 Map::Map () [inline]

Constructor.

Definition at line 109 of file Make\_map.H.

#### 10.155.2.2 Map::~~Map () [inline]

Destructor.

Definition at line 113 of file Make\_map.H.

### 10.155.3 Member Function Documentation

#### 10.155.3.1 void Map::create\_Map (char \* *file*, char *c*, Map \* *object*, Map \* *subject*, Map \* *action*)

Reads from the XML file and makes the map.

**Parameters:**

*file* enter the XML file name

*c* enter the type(o/s/a)

*object* enter the pointer to object ontology

*object* enter the pointer to subject ontology

*object* enter the pointer to action ontology

Definition at line 44 of file Make\_map.C.

References print\_map().

Referenced by Ontology::Ontology().

**10.155.3.2 void Map::delete\_element (std::string name) [inline]**

Delete an element given the key string.

**Parameters:**

*name* enter the name of entity to be retrieved

Definition at line 121 of file Make\_map.H.

Referenced by Ontology::delete\_element().

**10.155.3.3 const Entity \* Map::get (std::string name) const [inline]**

Find the element given the key string; returns a pointer to the entity.

**Parameters:**

*name* enter the name of entity to be retrieved

Definition at line 128 of file Make\_map.H.

Referenced by Ontology::get().

**10.155.3.4 void Map::insert\_element (std::string name, Entity & entity) [inline]**

Insert an element into the map.

**Parameters:**

*name* enter name of entity to be inserted

*entity* enter a reference to the entity to be inserted

Definition at line 116 of file Make\_map.H.

Referenced by Ontology::insert\_element().

**10.155.3.5 void Map::print\_map (char \* file) const**

Print contents of map.

Definition at line 68 of file Make\_map.C.

References Entity::get\_flag(), Entity::get\_name(), and Entity::print\_entity().

Referenced by create\_Map().

The documentation for this class was generated from the following files:

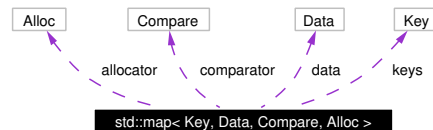
- [Make\\_map.H](#)
- [Make\\_map.C](#)

## 10.156 std::map< Key, Data, Compare, Alloc > Class Template Reference

The STL map container class. Use with `#include` .

```
#include <dummySTL.H>
```

Collaboration diagram for std::map< Key, Data, Compare, Alloc >:



### Public Attributes

- Key **keys**
- Data **data**
- Compare **comparator**
- Alloc **allocator**

#### 10.156.1 Detailed Description

```
template<class Key, class Data, class Compare, class Alloc> class std::map< Key,
Data, Compare, Alloc >
```

The STL map container class. Use with `#include` .

See [SGI's documentation](#) of std::map.

Definition at line 43 of file dummySTL.H.

The documentation for this class was generated from the following file:

- [dummySTL.H](#)

## 10.157 Matrix3 Class Reference

```
#include <Matrix3.H>
```

### Public Member Functions

- [Matrix3](#) ()
- [Matrix3](#) (const [Matrix3](#) &ot)
- void [setIdentity](#) ()
- void [setRotation](#) (double Yaw, double Pitch, double Roll)
- void [setRotationX](#) (double a)
- void [setRotationY](#) (double a)
- void [setRotationZ](#) (double a)
- [Matrix3](#) & [operator=](#) (const [Matrix3](#) &ot)
- [Matrix3](#) [operator \\*](#) (const [Matrix3](#) &ot) const
- void [operator \\*=](#) ([Matrix3](#) ot)
- bool [operator==](#) ([Matrix3](#) &ot) const
- bool [operator!=](#) ([Matrix3](#) &ot) const
- [Matrix3](#) [transpose](#) ()
- double [determinant](#) () const
- [Vector3](#) [transformVector](#) (const [Vector3](#) &V) const
- [Matrix3](#) [inverse](#) () const

### Public Attributes

- double [m](#) [3][3]

#### 10.157.1 Detailed Description

general : [Matrix3](#) is a class for handling 3x3 Matrix manipulation.

We do not use 4x4 matrix for view point changement as the calculus could be inefficent (some of the coefficients are null,  $m_{14} = m_{24} = m_{34} = 0$  et  $m_{44} = 1.0$  always).

Definition at line 35 of file [Matrix3.H](#).

#### 10.157.2 Constructor & Destructor Documentation

##### 10.157.2.1 [Matrix3::Matrix3](#) ()

default constructor : initialise to zero

Definition at line 26 of file [Matrix3.C](#).

### 10.157.2.2 Matrix3::Matrix3 (const Matrix3 & ot)

copy constructor

Definition at line 30 of file Matrix3.C.

## 10.157.3 Member Function Documentation

### 10.157.3.1 double Matrix3::determinant () const [inline]

get the determinant

Definition at line 103 of file Matrix3.H.

Referenced by inverse().

### 10.157.3.2 Matrix3 Matrix3::inverse () const

return inverse if it exists, otherwise identity

Definition at line 135 of file Matrix3.C.

References determinant(), m, and setIdentity().

### 10.157.3.3 Matrix3 Matrix3::operator \* (const Matrix3 & ot) const

multiplication with another matrix

Definition at line 98 of file Matrix3.C.

References m.

### 10.157.3.4 void Matrix3::operator \*= (Matrix3 ot)

operator \*=

Definition at line 116 of file Matrix3.C.

References m.

### 10.157.3.5 bool Matrix3::operator!= (Matrix3 & ot) const [inline]

comparison

Definition at line 81 of file Matrix3.H.

**10.157.3.6   [Matrix3](#) & Matrix3::operator= (const [Matrix3](#) & ot)**

copy operator

Definition at line 89 of file Matrix3.C.

References [m](#).

**10.157.3.7   bool Matrix3::operator== ([Matrix3](#) & ot) const   [inline]**

comparison

Definition at line 71 of file Matrix3.H.

References [m](#).

**10.157.3.8   void Matrix3::setIdentity ()**

Set the identity matrix

Definition at line 36 of file Matrix3.C.

Referenced by [inverse\(\)](#).

**10.157.3.9   void Matrix3::setRotation (double *Yaw*, double *Pitch*, double *Roll*)**

Set the matrice to rotation using yaw, pitch, roll angle

Definition at line 44 of file Matrix3.C.

**10.157.3.10   void Matrix3::setRotationX (double *a*)**

set the matrice to rotation around X

Definition at line 67 of file Matrix3.C.

**10.157.3.11   [Vector3](#) Matrix3::transformVector (const [Vector3](#) & V) const  
[inline]**

transforms a vector

Definition at line 113 of file Matrix3.H.

References [Vector3::x](#), [Vector3::y](#), and [Vector3::z](#).



**10.157.3.12** [Matrix3](#) `Matrix3::transpose()` [inline]

retrieves transpose

Definition at line 87 of file Matrix3.H.

References `m`.

**10.157.4 Member Data Documentation****10.157.4.1** `double` [Matrix3::m](#)[3][3]

we define the Matrix3 as 3 columns of 3 rows

Definition at line 39 of file Matrix3.H.

Referenced by `inverse()`, `operator*()`, `operator*=(())`, `operator=()`, `operator==(())`, and `transpose()`.

The documentation for this class was generated from the following files:

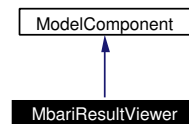
- Matrix3.H
- Matrix3.C

## 10.158 MbariResultViewer Class Reference

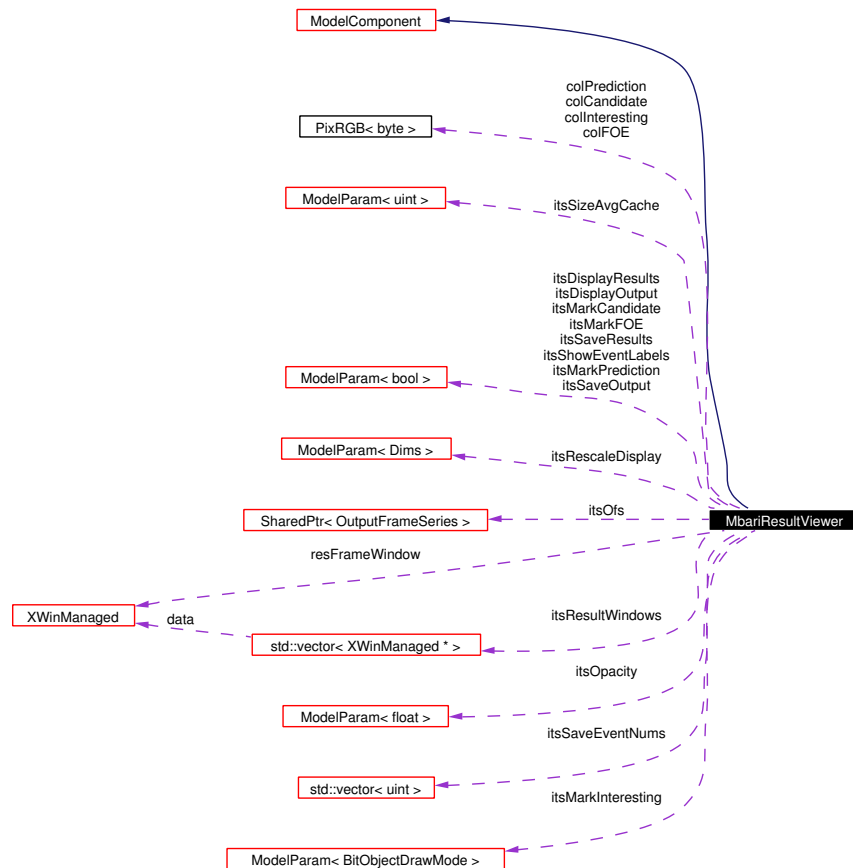
Class to facilitate displaying and saving of intermediate results.

```
#include <MbariResultViewer.H>
```

Inheritance diagram for MbariResultViewer:



Collaboration diagram for MbariResultViewer:



## Public Member Functions

- [MbariResultViewer](#) ([ModelManager](#) &mgr, [SharedPtr](#)< [OutputFrameSeries](#) > ofs)  
*Constructor - give a [ModelManager](#) and an [OutputFrameSeries](#).*
- [~MbariResultViewer](#) ()  
*Destructor - destroy all the windows that may be present.*
- virtual void [exportOptions](#) (const int whatToExport, const bool recurse)  
*implements [ModelComponent::exportOptions](#) - export out command line options*
- virtual bool [setModelParamString](#) (const [std::string](#) &name, const [std::string](#) &value, const bool recurse, const bool warn)  
*implements [ModelComponent::setModelParamString](#)*
- virtual void [reset](#) (const bool recurse)  
*implements [ModelComponent::reset](#) - destroy all the windows*
- template<class T> void [output](#) (const [Image](#)< T > &img, const [uint](#) frameNum, const [std::string](#) &resultName, const int resNum=-1)  
*save and/or display intermediate result*
- template<class T> void [display](#) (const [Image](#)< T > &img, const [uint](#) frameNum, const [std::string](#) &resultName, const int resNum=-1)  
*display intermediate result*
- void [save](#) (const [Image](#)< [PixRGB](#)< [byte](#) > > &img, const [uint](#) frameNum, const [std::string](#) &resultName, const int resNum=-1)  
*save intermediate result*
- void [save](#) (const [Image](#)< [byte](#) > &img, const [uint](#) frameNum, const [std::string](#) &resultName, const int resNum=-1)  
*save intermediate result*
- void [save](#) (const [Image](#)< [float](#) > &img, const [uint](#) frameNum, const [std::string](#) &resultName, const int resNum=-1)  
*save intermediate result*
- void [outputResultFrame](#) (const [Image](#)< [PixRGB](#)< [byte](#) > > &resultImg, const [uint](#) frameNum, [VisualEventSet](#) &evts, [PropertyVectorSet](#) &pvs, const int circleRadius)  
*save and possibly display an output frame*

- bool [isLoadEventsNameSet](#) () const  
*check if the LoadEventsName is set*
- void [loadVisualEventSet](#) ([VisualEventSet](#) &ves) const  
*load the [VisualEventSet](#) from the file LoadEventsName*
- bool [isLoadPropertiesNameSet](#) () const  
*check if the LoadPropertiesName is set*
- void [loadProperties](#) ([PropertyVectorSet](#) &pvs) const  
*load the PropertyVector from the file LoadPropertiesName*
- bool [isSaveEventsNameSet](#) () const  
*check if the SaveEventsName is set*
- void [saveVisualEventSet](#) (const [VisualEventSet](#) &ves) const  
*save the [VisualEventSet](#) to the file SaveEventsName*
- bool [isSavePropertiesNameSet](#) () const  
*check if the SavePropertiesName is set*
- void [saveProperties](#) (const [PropertyVectorSet](#) &pvs) const  
*save the PropertyVector to the file SavePropertiesName*
- bool [isSavePositionsNameSet](#) () const  
*check if the SavePositionsName is set*
- void [savePositions](#) (const [VisualEventSet](#) &ves) const  
*save the positions to the file SavePositionsName*
- bool [needFrames](#) () const  
*determine whether the frames actually need to be loaded*
- uint [getAvgCacheSize](#) () const  
*returns the size of the cache for running averages (from the command line)*
- bool [isSaveEventClip](#) () const  
*determine whether we have to save any event clips*
- uint [numSaveEventClips](#) () const  
*return the number of event clips we have to save*

- `uint getSaveEventClipNum (uint idx) const`  
*returns the event number with idx to be saved as a clip*
- `void saveSingleEventFrame (const Image< PixRGB< byte > > &img, int frameNum, const VisualEvent &event)`  
*save a cut-out piece of frame containing a single event*

## Protected Member Functions

- `void freeMem ()`  
*destroy windows and other internal variables*
- `uint getNumFromString (const std::string &resultName)`  
*get an internal reference number from a resultName*
- `std::string getLabel (const uint num, const uint frameNum, const int resNum=-1)`  
*make a label text from the reference number and the frame number*
- `std::string getFileStem (const std::string &resultName, const int resNum=-1)`  
*make the file stem for saving files*
- `template<class T> XWinManaged * displayImage (const Image< T > &img, XWinManaged *win, const char *label)`  
*draw an image to a window, deal with creating and rescaling the window*
- `void parseSaveEventNums (const std::string &value)`  
*parse the SaveEventNumString and store the numbers in itsSaveEventNums*

## Protected Attributes

- `ModelParam< bool > itsSaveResults`  
*whether intermediate results are saved*
- `ModelParam< bool > itsDisplayResults`  
*whether intermediate results are displayed*
- `ModelParam< BitObjectDrawMode > itsMarkInteresting`  
*the way interesting events are marked*

- [ModelParam< float > itsOpacity](#)  
*opacity of shape or outline markings of events*
- [ModelParam< bool > itsMarkCandidate](#)  
*whether candidates for int. events are marked*
- [ModelParam< bool > itsMarkPrediction](#)  
*whether predictions of event locations are marked*
- [ModelParam< bool > itsMarkFOE](#)  
*whether the focus of expansion is marked*
- [ModelParam< bool > itsSaveOutput](#)  
*whether the output frames are saved*
- [ModelParam< bool > itsDisplayOutput](#)  
*whether the output frames are displayed*
- [ModelParam< bool > itsShowEventLabels](#)  
*whether the event labels are written into output frames*
- [ModelParam< Dims > itsRescaleDisplay](#)  
*rescale intermediate results to this scale for display*
- [ModelParam< uint > itsSizeAvgCache](#)  
*the number of images used for running average*
- [ModelParam< std::string > itsSaveEventsName](#)
- [ModelParam< std::string > itsLoadEventsName](#)
- [ModelParam< std::string > itsSavePropertiesName](#)
- [ModelParam< std::string > itsLoadPropertiesName](#)
- [ModelParam< std::string > itsSavePositionsName](#)
- [ModelParam< std::string > itsSaveEventNumString](#)

### 10.158.1 Detailed Description

Class to facilitate displaying and saving of intermediate results.

This class provides command line options to control the saving and displaying of intermediate results as well as output frames for MBARI programs

Definition at line 62 of file MbariResultViewer.H.

## 10.158.2 Constructor & Destructor Documentation

### 10.158.2.1 MbariResultViewer::MbariResultViewer ([ModelManager](#) & *mgr*, [SharedPtr](#)< [OutputFrameSeries](#) > *ofs*)

Constructor - give a [ModelManager](#) and an [OutputFrameSeries](#).

If the [OutputFrameSeries](#) is not yet registered in the manager, it will be registered here

Definition at line 51 of file MbariResultViewer.C.

References [BODMbbox](#), [ModelComponent::hasSubComponent\(\)](#), and [ModelManager::registerComponent\(\)](#).

### 10.158.2.2 MbariResultViewer::~MbariResultViewer ()

Destructor - destroy all the windows that may be present.

Definition at line 85 of file MbariResultViewer.C.

References [freeMem\(\)](#).

## 10.158.3 Member Function Documentation

### 10.158.3.1 `template<class T> void MbariResultViewer::display (const Image< T > & img, const uint frameNum, const std::string & resultName, const int resNum = -1)`

display intermediate result

#### Parameters:

*img* the image containing the intermediate result

*frameNum* the frame number of the image

*resultName* a string that defines the kind of interemediate result - this is appended to the filestem for the file name and/or window title

*resNum* if there are several results of this type for each frame, the number of the resul

Definition at line 202 of file MbariResultViewer.C.

References [displayImage\(\)](#), [getLabel\(\)](#), [getNumFromString\(\)](#), and [uint](#).

Referenced by [output\(\)](#).

**10.158.3.2** `template<class T> XWinManaged * MbariResultViewer::display-  
Image (const Image< T > & img, XWinManaged * win, const char *  
label)` [protected]

draw an image to a window, deal with creating and rescaling the window

Definition at line 411 of file MbariResultViewer.C.

References [XWinManaged::drawImage\(\)](#), [XWindow::getDims\(\)](#), [Image< T >::getDims\(\)](#), [ModelParam< Dims >::getVal\(\)](#), [Dims::isEmpty\(\)](#), [itsRescaleDisplay](#), [rescale\(\)](#), and [XWindow::setTitle\(\)](#).

Referenced by [display\(\)](#), and [outputResultFrame\(\)](#).

**10.158.3.3** `void MbariResultViewer::exportOptions (const int whatToExport,  
const bool recurse)` [virtual]

implements [ModelComponent::exportOptions](#) - export out command line options

Reimplemented from [ModelComponent](#).

Definition at line 92 of file MbariResultViewer.C.

References [ModelComponent::exportOptions\(\)](#), [itsDisplayOutput](#), [itsDisplayResults](#), [itsMarkCandidate](#), [itsMarkFOE](#), [itsMarkInteresting](#), [itsMarkPrediction](#), [itsOpacity](#), [itsRescaleDisplay](#), [itsSaveOutput](#), [itsShowEventLabels](#), [itsSizeAvgCache](#), [OPTEXP\\_MRV](#), and [ModelManager::requestOption\(\)](#).

**10.158.3.4** `void MbariResultViewer::freeMem ()` [protected]

destroy windows and other internal variables

Definition at line 175 of file MbariResultViewer.C.

References [uint](#).

Referenced by [reset\(\)](#), and [~MbariResultViewer\(\)](#).

**10.158.3.5** `uint MbariResultViewer::getAvgCacheSize ()` const

returns the size of the cache for running averages (from the command line)

Definition at line 348 of file MbariResultViewer.C.

References [ModelParam< uint >::getVal\(\)](#), [itsSizeAvgCache](#), and [uint](#).



**10.158.3.6** `std::string MbariResultViewer::getFileStem (const std::string & resultName, const int resNum = -1)` [protected]

make the file stem for saving files

Definition at line 398 of file MbariResultViewer.C.

Referenced by `getLabel()`, and `save()`.

**10.158.3.7** `std::string MbariResultViewer::getLabel (const uint num, const uint frameNum, const int resNum = -1)` [protected]

make a label text from the reference number and the frame number

Definition at line 388 of file MbariResultViewer.C.

References `ASSERT`, `getFileStem()`, and `uint`.

Referenced by `display()`.

**10.158.3.8** `uint MbariResultViewer::getNumFromString (const std::string & resultName)` [protected]

get an internal reference number from a resultName

If this resultName is new, a new reference is created

Definition at line 373 of file MbariResultViewer.C.

References `uint`.

Referenced by `display()`.

**10.158.3.9** `uint MbariResultViewer::getSaveEventClipNum (uint idx) const`

returns the event number with idx to be saved as a clip

Definition at line 366 of file MbariResultViewer.C.

References `ASSERT`, and `uint`.

**10.158.3.10** `bool MbariResultViewer::isLoadEventsNameSet () const`

check if the LoadEventsName is set

Definition at line 270 of file MbariResultViewer.C.

Referenced by `needFrames()`.

**10.158.3.11 bool MbariResultViewer::isLoadPropertiesNameSet () const**

check if the LoadPropertiesName is set

Definition at line 284 of file MbariResultViewer.C.

**10.158.3.12 bool MbariResultViewer::isSaveEventClip () const**

determine whether we have to save any event clips

Definition at line 354 of file MbariResultViewer.C.

Referenced by needFrames().

**10.158.3.13 bool MbariResultViewer::isSaveEventsNameSet () const**

check if the SaveEventsName is set

Definition at line 298 of file MbariResultViewer.C.

**10.158.3.14 bool MbariResultViewer::isSavePositionsNameSet () const**

check if the SavePositionsName is set

Definition at line 326 of file MbariResultViewer.C.

**10.158.3.15 bool MbariResultViewer::isSavePropertiesNameSet () const**

check if the SavePropertiesName is set

Definition at line 312 of file MbariResultViewer.C.

**10.158.3.16 void MbariResultViewer::loadProperties ([PropertyVectorSet](#) & pvs) const**

load the PropertyVector from the file LoadPropertiesName

Definition at line 290 of file MbariResultViewer.C.

References [PropertyVectorSet::readFromStream\(\)](#).

**10.158.3.17 void MbariResultViewer::loadVisualEventSet ([VisualEventSet](#) & ves) const**

load the [VisualEventSet](#) from the file LoadEventsName

Definition at line 277 of file MbariResultViewer.C.

References VisualEventSet::readFromStream().

### 10.158.3.18 bool MbariResultViewer::needFrames () const

determine whether the frames actually need to be loaded

Definition at line 340 of file MbariResultViewer.C.

References ModelParam< bool >::getVal(), isLoadEventsNameSet(), isSaveEventClip(), itsDisplayOutput, and itsSaveOutput.

### 10.158.3.19 uint MbariResultViewer::numSaveEventClips () const

return the number of event clips we have to save

Definition at line 360 of file MbariResultViewer.C.

References uint.

### 10.158.3.20 template<class T> void MbariResultViewer::output (const Image< T > & img, const uint frameNum, const std::string & resultName, const int resNum = -1)

save and/or display intermediate result

whether the result image is saved and/or displayed is determined by the ModelParams

#### Parameters:

**img** the image containing the intermediate result

**frameNum** the frame number of the image

**resultName** a string that defines the kind of interemediate result - this is appended to the filestem for the file name and/or window title

**resNum** if there are several results of this type for each frame, the number of the result

Definition at line 193 of file MbariResultViewer.C.

References display(), ModelParam< bool >::getVal(), itsDisplayResults, save(), and uint.

**10.158.3.21** `void MbariResultViewer::outputResultFrame (const Image< PixRGB< byte > > & resultImg, const uint frameNum, VisualEventSet & evts, PropertyVectorSet & pvs, const int circleRadius)`

save and possibly display an output frame

Depending on the ModelParams, interesting events, candidates for interesting events and skipped event locations are marked in the frame, the frame is displayed (if desired) and saved.

**Parameters:**

*resultImg* the output image

*frameNum* the frame number of the output image

*evts* the event set to be used for drawing the events

*circleRadius* the radius of the circles used for marking

Definition at line 241 of file MbariResultViewer.C.

References [displayImage\(\)](#), [VisualEventSet::drawTokens\(\)](#), [ModelParam](#)< float >::getVal(), [ModelParam](#)< BitObjectDrawMode >::getVal(), [ModelParam](#)< bool >::getVal(), [itsDisplayOutput](#), [itsMarkInteresting](#), [itsOpacity](#), [itsSaveOutput](#), [itsShowEventLabels](#), and [uint](#).

**10.158.3.22** `void MbariResultViewer::parseSaveEventNums (const std::string & value)` [protected]

parse the SaveEventNumString and store the numbers in itsSaveEventNums

Definition at line 488 of file MbariResultViewer.C.

References [uint](#).

Referenced by [setModelParamString\(\)](#).

**10.158.3.23** `void MbariResultViewer::reset (const bool recurse)` [virtual]

implements [ModelComponent::reset](#) - destroy all the windows

Reimplemented from [ModelComponent](#).

Definition at line 165 of file MbariResultViewer.C.

References [freeMem\(\)](#), and [ModelComponent::reset\(\)](#).

**10.158.3.24** `void MbariResultViewer::save (const Image< float > & img, const uint frameNum, const std::string & resultName, const int resNum = -1)`

save intermediate result

**Parameters:**

*img* the image containing the intermediate result

*frameNum* the frame number of the image

*resultName* a string that defines the kind of interemediate result - this is appended to the filestem for the file name and/or window title

*resNum* if there are several results of this type for each frame, the number of the resul

Definition at line 231 of file MbariResultViewer.C.

References `getFileStem()`, and `uint`.

**10.158.3.25** `void MbariResultViewer::save (const Image< byte > & img, const uint frameNum, const std::string & resultName, const int resNum = -1)`

save intermediate result

**Parameters:**

*img* the image containing the intermediate result

*frameNum* the frame number of the image

*resultName* a string that defines the kind of interemediate result - this is appended to the filestem for the file name and/or window title

*resNum* if there are several results of this type for each frame, the number of the resul

Definition at line 221 of file MbariResultViewer.C.

References `getFileStem()`, and `uint`.

**10.158.3.26** `void MbariResultViewer::save (const Image< PixRGB< byte > > & img, const uint frameNum, const std::string & resultName, const int resNum = -1)`

save intermediate result

**Parameters:**

*img* the image containing the intermediate result

***frameNum*** the frame number of the image

***resultName*** a string that defines the kind of interemmediate result - this is appended to the filestem for the file name and/or window title

***resNum*** if there are several results of this type for each frame, the number of the result

Definition at line 211 of file MbariResultViewer.C.

References `getFileStem()`, and `uint`.

Referenced by `output()`.

#### 10.158.3.27 **void MbariResultViewer::savePositions (const VisualEventSet & ves) const**

save the positions to the file `SavePositionsName`

Definition at line 332 of file MbariResultViewer.C.

References `VisualEventSet::writePositions()`.

#### 10.158.3.28 **void MbariResultViewer::saveProperties (const PropertyVectorSet & pvs) const**

save the `PropertyVector` to the file `SavePropertiesName`

Definition at line 318 of file MbariResultViewer.C.

References `PropertyVectorSet::writeToStream()`.

#### 10.158.3.29 **void MbariResultViewer::saveSingleEventFrame (const Image< PixRGB< byte > > & img, int frameNum, const VisualEvent & event)**

save a cut-out piece of frame containing a single event

Definition at line 447 of file MbariResultViewer.C.

References `ASSERT`, `Token::bitObject`, `crop()`, `BitObject::getBoundingBox()`, `VisualEvent::getEventNum()`, `VisualEvent::getMaxObjectDims()`, `VisualEvent::getToken()`, `Dims::h()`, `Rectangle::height()`, `VisualEvent::isFrameOk()`, `Rectangle::left()`, `Rectangle::top()`, `Dims::w()`, and `Rectangle::width()`.

#### 10.158.3.30 **void MbariResultViewer::saveVisualEventSet (const VisualEventSet & ves) const**

save the `VisualEventSet` to the file `SaveEventsName`

Definition at line 304 of file MbariResultViewer.C.

References [VisualEventSet::writeToStream\(\)](#).

**10.158.3.31** `bool MbariResultViewer::setModelParamString (const std::string & name, const std::string & value, const bool recurse, const bool warn) [virtual]`

implements [ModelComponent::setModelParamString](#)

intercepts setting of MRVmarkXXX variables and sets drawing colors accordingly

Reimplemented from [ModelComponent](#).

Definition at line 121 of file MbariResultViewer.C.

References [ModelParamBase::getName\(\)](#), [ModelParam< bool >::getVal\(\)](#), [itsMarkCandidate](#), [itsMarkFOE](#), [itsMarkPrediction](#), [parseSaveEventNums\(\)](#), and [ModelComponent::setModelParamString\(\)](#).

## 10.158.4 Member Data Documentation

**10.158.4.1** `ModelParam<bool> MbariResultViewer::itsDisplayOutput [protected]`

whether the output frames are displayed

Definition at line 242 of file MbariResultViewer.H.

Referenced by [exportOptions\(\)](#), [needFrames\(\)](#), and [outputResultFrame\(\)](#).

**10.158.4.2** `ModelParam<bool> MbariResultViewer::itsDisplayResults [protected]`

whether intermediate results are displayed

Definition at line 235 of file MbariResultViewer.H.

Referenced by [exportOptions\(\)](#), and [output\(\)](#).

**10.158.4.3** `ModelParam<bool> MbariResultViewer::itsMarkCandidate [protected]`

whether candidates for int. events are marked

Definition at line 238 of file MbariResultViewer.H.

Referenced by [exportOptions\(\)](#), and [setModelParamString\(\)](#).

**10.158.4.4** [\*\*ModelParam\*\*<bool> \*\*MbariResultViewer::itsMarkFOE\*\*](#)  
[protected]

whether the focus of expansion is marked

Definition at line 240 of file MbariResultViewer.H.

Referenced by exportOptions(), and setModelParamString().

**10.158.4.5** [\*\*ModelParam\*\*<BitObjectDrawMode> \*\*MbariResultViewer::itsMarkInteresting\*\*](#)  
[protected]

the way interesting events are marked

Definition at line 236 of file MbariResultViewer.H.

Referenced by exportOptions(), and outputResultFrame().

**10.158.4.6** [\*\*ModelParam\*\*<bool> \*\*MbariResultViewer::itsMarkPrediction\*\*](#)  
[protected]

whether predictions of event locations are marked

Definition at line 239 of file MbariResultViewer.H.

Referenced by exportOptions(), and setModelParamString().

**10.158.4.7** [\*\*ModelParam\*\*<float> \*\*MbariResultViewer::itsOpacity\*\*](#)  
[protected]

opacity of shape or outline markings of events

Definition at line 237 of file MbariResultViewer.H.

Referenced by exportOptions(), and outputResultFrame().

**10.158.4.8** [\*\*ModelParam\*\*<Dims> \*\*MbariResultViewer::itsRescaleDisplay\*\*](#)  
[protected]

rescale intermediate results to this scale for display

Definition at line 245 of file MbariResultViewer.H.

Referenced by displayImage(), and exportOptions().



**10.158.4.9** [ModelParam<bool> MbariResultViewer::itsSaveOutput](#)  
[protected]

whether the output frames are saved

Definition at line 241 of file MbariResultViewer.H.

Referenced by exportOptions(), needFrames(), and outputResultFrame().

**10.158.4.10** [ModelParam<bool> MbariResultViewer::itsSaveResults](#)  
[protected]

whether intermediate results are saved

Definition at line 234 of file MbariResultViewer.H.

**10.158.4.11** [ModelParam<bool> MbariResultViewer::itsShowEventLabels](#)  
[protected]

whether the event labels are written into output frames

Definition at line 243 of file MbariResultViewer.H.

Referenced by exportOptions(), and outputResultFrame().

**10.158.4.12** [ModelParam<uint> MbariResultViewer::itsSizeAvgCache](#)  
[protected]

the number of images used for running average

Definition at line 247 of file MbariResultViewer.H.

Referenced by exportOptions(), and getAvgCacheSize().

The documentation for this class was generated from the following files:

- [MbariResultViewer.H](#)
- [MbariResultViewer.C](#)

## 10.159 Raster::MetaData Class Reference

Metadata information (typically obtained from GPS).

```
#include <Raster.H>
```

### Public Member Functions

- [MetaData \(\)](#)  
*Constructor.*
- [MetaData \(const MetaData &m\)](#)  
*Constructor.*
- [MetaData \(std::string s\)](#)  
*Constructor.*
- void [parseMetaData \(std::string s\)](#)  
*Parse from string.*
- float [getAccel \(\)](#) const  
*Access function.*
- float [getSteer \(\)](#) const  
*Access function.*
- float [getLat \(\)](#) const  
*Access function.*
- float [getLon \(\)](#) const  
*Access function.*
- float [getEV \(\)](#) const  
*Access function.*
- float [getNV \(\)](#) const  
*Access function.*

## Public Attributes

- float [accel](#)  
*Acceleration.*
- float [steer](#)  
*Steering.*
- float [lat](#)  
*Latitude.*
- float [lon](#)  
*Longitude.*
- float [ev](#)  
*Velocity in east direction.*
- float [nv](#)  
*Velocity in north direction.*

### 10.159.1 Detailed Description

Metadata information (typically obtained from GPS).

Definition at line 79 of file Raster.H.

### 10.159.2 Constructor & Destructor Documentation

#### 10.159.2.1 Raster::MetaData::MetaData ()

Constructor.

Definition at line 351 of file Raster.C.

#### 10.159.2.2 Raster::MetaData::MetaData (const [MetaData](#) & m)

Constructor.

Definition at line 355 of file Raster.C.

**10.159.2.3 Raster::MetaData::MetaData (std::string s)**

Constructor.

Definition at line 360 of file Raster.C.

References `parseMetaData()`.

**10.159.3 Member Function Documentation****10.159.3.1 float Raster::MetaData::getAccel () const**

Access function.

Definition at line 393 of file Raster.C.

References `accel`.

**10.159.3.2 float Raster::MetaData::getEV () const**

Access function.

Definition at line 405 of file Raster.C.

References `ev`.

**10.159.3.3 float Raster::MetaData::getLat () const**

Access function.

Definition at line 399 of file Raster.C.

**10.159.3.4 float Raster::MetaData::getLon () const**

Access function.

Definition at line 402 of file Raster.C.

**10.159.3.5 float Raster::MetaData::getNV () const**

Access function.

Definition at line 408 of file Raster.C.

References `nv`.

**10.159.3.6 float Raster::MetaData::getSteer () const**

Access function.

Definition at line 396 of file Raster.C.

References `steer`.

**10.159.3.7 void Raster::MetaData::parseMetaData (std::string s)**

Parse from string.

Definition at line 366 of file Raster.C.

References `accel`, `ev`, `nv`, and `steer`.

Referenced by `MetaData()`.

**10.159.4 Member Data Documentation****10.159.4.1 float Raster::MetaData::accel**

Acceleration.

Definition at line 113 of file Raster.H.

Referenced by `getAccel()`, and `parseMetaData()`.

**10.159.4.2 float Raster::MetaData::ev**

Velocity in east direction.

Definition at line 117 of file Raster.H.

Referenced by `getEV()`, and `parseMetaData()`.

**10.159.4.3 float Raster::MetaData::lat**

Latitude.

Definition at line 115 of file Raster.H.

**10.159.4.4 float Raster::MetaData::lon**

Longitude.

Definition at line 116 of file Raster.H.

**10.159.4.5   float [Raster::MetaData::nv](#)**

Velocity in north direction.

Definition at line 118 of file Raster.H.

Referenced by [getNV\(\)](#), and [parseMetaData\(\)](#).

**10.159.4.6   float [Raster::MetaData::steer](#)**

Steering.

Definition at line 114 of file Raster.H.

Referenced by [getSteer\(\)](#), and [parseMetaData\(\)](#).

The documentation for this class was generated from the following files:

- [Raster.H](#)
- [Raster.C](#)

## 10.160 MexBuf Class Reference

This streambuf is used to re-direct cout and cerr.

```
#include <mexLog.H>
```

### Public Member Functions

- virtual int **overflow** (int c)
- virtual std::streamsize **xspn** (const char \*s, const std::streamsize n)

#### 10.160.1 Detailed Description

This streambuf is used to re-direct cout and cerr.

Definition at line 59 of file mexLog.H.

The documentation for this class was generated from the following file:

- [mexLog.H](#)

## 10.161 MexBufInit Class Reference

This class triggers the redirection of the standard streams at construction.

```
#include <mexLog.H>
```

### Public Member Functions

- **MexBufInit** ([MexBuf](#) &buf)

#### 10.161.1 Detailed Description

This class triggers the redirection of the standard streams at construction.

Definition at line 84 of file mexLog.H.

The documentation for this class was generated from the following file:

- [mexLog.H](#)

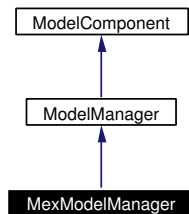


## 10.162 MexModelManager Class Reference

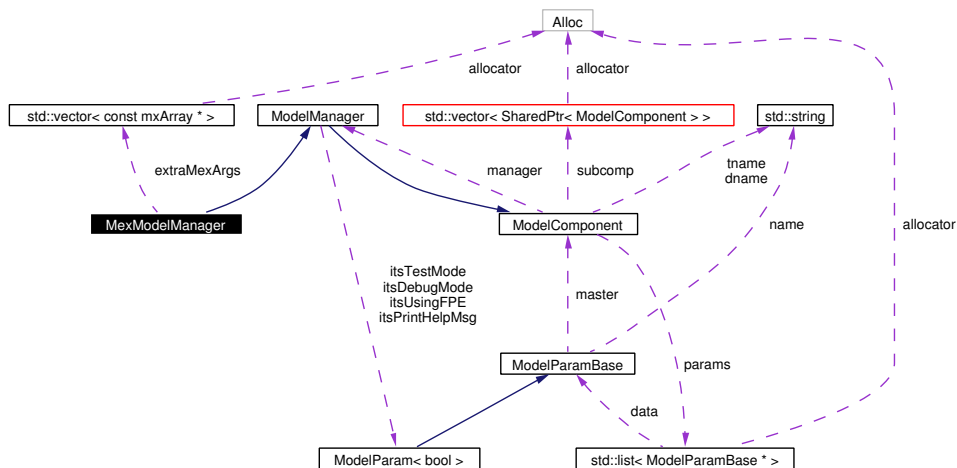
A special form of [ModelManager](#) used in mex files.

```
#include <MexModelManager.H>
```

Inheritance diagram for MexModelManager:



Collaboration diagram for MexModelManager:



### Public Member Functions

- [MexModelManager](#) (const [std::string](#) &descrName="Mex Model", const [std::string](#) &tagName="mexModel")  
*Default constructor.*
- virtual [~MexModelManager](#) ()

*Virtual destructor ensures proper destruction of derived classes.*

- `bool usingFPE () const`  
*returns false - no floating-point exceptions allowed in mex files!*
- `bool parseCommandLine (const int nrhs, const mxArray *prhs[], const char *usage, const int minarg, const int maxarg)`  
*Parse all input arguments.*
- `uint numExtraMexArgs () const`  
*Get the number of available extra args.*
- `const mxArray * getExtraMexArg (const uint num) const`  
*Get an extra arg, by number.*
- `void printUsage (const char *usage)`  
*Print short usage message.*

## Protected Member Functions

- `virtual void exitProgram (int return_code=0)`  
*Called to leave the program, e.g. when printing the Help messages.*

## Protected Attributes

- `std::vector< const mxArray * > extraMexArgs`  
*non-option command-line args*

### 10.162.1 Detailed Description

A special form of `ModelManager` used in mex files.

Definition at line 44 of file `MexModelManager.H`.

## 10.162.2 Constructor & Destructor Documentation

### 10.162.2.1 MexModelManager::MexModelManager (const [std::string](#) & *descrName* = "Mex Model", const [std::string](#) & *tagName* = "mexModel")

Default constructor.

**Parameters:**

*descrName* descriptive name

*tagName* tag name (see [ModelComponent.H](#))

Definition at line 43 of file MexModelManager.C.

References [fpExceptionsLock\(\)](#), [fpExceptionsOff\(\)](#), [fpExceptionsUnlock\(\)](#), [ModelParam< bool >::setVal\(\)](#), and [ModelManager::unRequestOption\(\)](#).

### 10.162.2.2 MexModelManager::~MexModelManager () [virtual]

Virtual destructor ensures proper destruction of derived classes.

Definition at line 56 of file MexModelManager.C.

## 10.162.3 Member Function Documentation

### 10.162.3.1 void MexModelManager::exitProgram (int *return\_code* = 0) [protected, virtual]

Called to leave the program, e.g. when printing the Help messages.

This implementation calls [mexErrMsgTxt](#).

Reimplemented from [ModelManager](#).

Definition at line 173 of file MexModelManager.C.

References [mexFatal\(\)](#).

### 10.162.3.2 const [mxArray](#) \* MexModelManager::getExtraMexArg (const [uint](#) *num*) const

Get an extra arg, by number.

Definition at line 156 of file MexModelManager.C.

References [extraMexArgs](#), [mexFatal\(\)](#), and [uint](#).

**10.162.3.3   [uint](#) MexModelManager::numExtraMexArgs () const**

Get the number of available extra args.

Definition at line 152 of file MexModelManager.C.

References [extraMexArgs](#), and [uint](#).

**10.162.3.4   [bool](#) MexModelManager::parseCommandLine (const int *nrhs*, const mxArray \* *prhs*[], const char \* *usage*, const int *minarg*, const int *maxarg*)**

Parse all input arguments.

All strings that start with '-' are interpreted as command-line options, all other arguments will be available though [getExtraMexArg\(\)](#). Use '-help' for a list of available options. Returns true upon fully successful parse.

**Parameters:**

***nrhs*** the number of command-line args

***prhs*** the input args

***usage*** a textual description of the expected usage format

***minarg*** the minimum number of non-optional arguments

***maxarg*** the maximum number of non-optional args, or -1 if unbounded

Definition at line 64 of file MexModelManager.C.

References [ModelManager::exportOptions\(\)](#), [extraMexArgs](#), [mexError\(\)](#), [OPTEXP\\_ALL](#), [printUsage\(\)](#), and [uint](#).

**10.162.3.5   [void](#) MexModelManager::printUsage (const char \* *usage*)**

Print short usage message.

This is automatically called during [parseCommandLine\(\)](#) if necessary

Definition at line 166 of file MexModelManager.C.

References [mexFatal\(\)](#).

Referenced by [parseCommandLine\(\)](#).

**10.162.3.6   [bool](#) MexModelManager::usingFPE () const**

returns false - no floating-point exceptions allowed in mex files!

Reimplemented from [ModelManager](#).

Definition at line 60 of file MexModelManager.C.

## 10.162.4 Member Data Documentation

### 10.162.4.1 [std::vector<const mxArray\\*>](#) [MexModelManager::extraMexArgs](#) [protected]

non-option command-line args

Definition at line 92 of file MexModelManager.H.

Referenced by [getExtraMexArg\(\)](#), [numExtraMexArgs\(\)](#), and [parseCommandLine\(\)](#).

The documentation for this class was generated from the following files:

- [MexModelManager.H](#)
- [MexModelManager.C](#)

## 10.163 MexReturn Class Reference

Class to help manage return values in mex files.

```
#include <mexConverts.H>
```

### Public Member Functions

- [MexReturn](#) (int nlhs, mxArray \*plhs[ ])   
*Constructor takes arguments from the mexFunction call.*
- virtual [~MexReturn](#) ()   
*Destructor checks that all arguments are assigned.*
- int [numArgs](#) ()   
*returns the number of return arguments*
- bool [isNumOK](#) (int num)   
*test if  $0 \leq \text{num} < \text{numArgs}()$*
- bool [store](#) (int num, mxArray \*val)   
*store a mxArray as return argumnt num; true if successful*
- bool [store](#) (int num, double val)   
*store double scalar*
- template<class T> bool [store](#) (int num, const [std::vector](#)< T > &val)   
*store a vector of numbers*
- template<class T> bool [store](#) (int num, const [Image](#)< T > &val)   
*store a grayscale image*
- template<class T> bool [store](#) (int num, const [Image](#)< [PixRGB](#)< T > > &val)   
*store an RGB image*

### 10.163.1 Detailed Description

Class to help manage return values in mex files.

Definition at line 138 of file mexConverts.H.

## 10.163.2 Constructor & Destructor Documentation

### 10.163.2.1 MexReturn::MexReturn (int *nlhs*, mxArray \* *plhs*[ ])

Constructor takes arguments from the mexFunction call.

### 10.163.2.2 MexReturn::~MexReturn () [virtual]

Destructor checks that all arguments are assigned.

Definition at line 508 of file mexConverts.C.

References `mexError()`, and `mexFatal()`.

## 10.163.3 Member Function Documentation

### 10.163.3.1 bool MexReturn::isNumOK (int *num*)

test if  $0 \leq \text{num} < \text{numArgs}()$

Definition at line 526 of file mexConverts.C.

References `numArgs()`.

Referenced by `store()`.

### 10.163.3.2 int MexReturn::numArgs ()

returns the number of return arguments

Definition at line 522 of file mexConverts.C.

Referenced by `isNumOK()`.

### 10.163.3.3 template<class T> bool MexReturn::store (int *num*, const Image<PixRGB< T > > & *val*)

store an RGB image

Definition at line 559 of file mexConverts.C.

References `isNumOK()`, `RGBImage2mexArray()`, and `store()`.

### 10.163.3.4 template<class T> bool MexReturn::store (int *num*, const Image< T > & *val*)

store a grayscale image

Definition at line 551 of file mexConverts.C.

References [Image2mexArray\(\)](#), [isNumOK\(\)](#), and [store\(\)](#).

#### **10.163.3.5    `template<class T> bool MexReturn::store (int num, const std::vector< T > & val)`**

store a vector of numbers

Definition at line 543 of file mexConverts.C.

References [isNumOK\(\)](#), [store\(\)](#), and [Vector2mexArr\(\)](#).

#### **10.163.3.6    `bool MexReturn::store (int num, double val)`**

store double scalar

Definition at line 538 of file mexConverts.C.

References [store\(\)](#).

#### **10.163.3.7    `bool MexReturn::store (int num, mxArray * val)`**

store a mxArray as return argumnt num; true if successful

Definition at line 530 of file mexConverts.C.

References [isNumOK\(\)](#).

Referenced by [store\(\)](#).

The documentation for this class was generated from the following files:

- [mexConverts.H](#)
- [mexConverts.C](#)

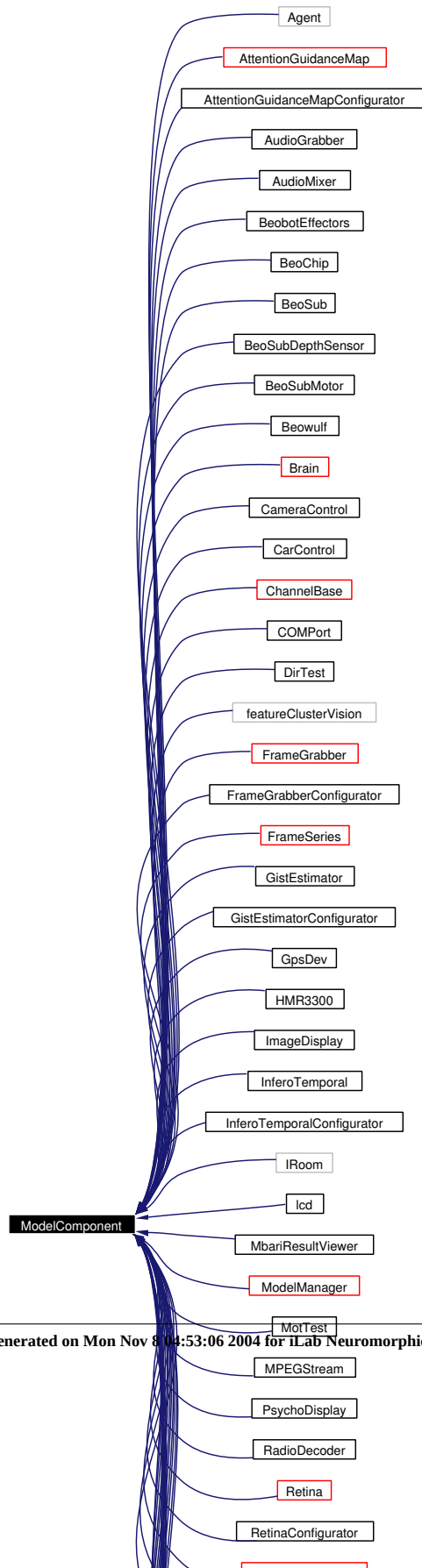


## 10.164 ModelComponent Class Reference

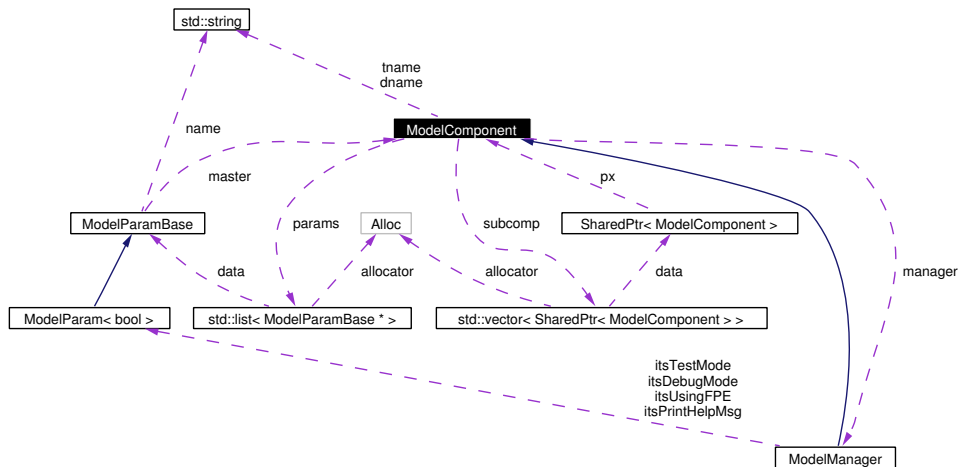
Base class for parameterized model components.

```
#include <ModelComponent.H>
```

Inheritance diagram for ModelComponent:



Collaboration diagram for ModelComponent:



## Public Member Functions

### Constructors and Destructors

- **ModelComponent** (**ModelManager** &mgr, const **std::string** &descrName, const **std::string** &tagName)  
*Constructor.*
- virtual **~ModelComponent** ()  
*Virtual destructor ensures proper destruction of derived classes.*
- void **start** ()  
*Start the component.*
- bool **started** () const  
*Did we already get a call to **start()**?*
- void **stop** ()  
*Stop the component.*
- void **destroyManager** ()  
*Signify that our **ModelManager** just got destroyed.*

### Name functions

- **std::string** **descriptiveName** () const

*Get a human-readable descriptive name for this component.*

- void `setDescriptionName` (const `std::string` &name)  
*Change this component's descriptive name.*
- `std::string tagName` () const  
*Get a programmer-usable tag name for this object.*
- void `setTagName` (const `std::string` &name)  
*Change this component's tag name.*

### Sub-Component management functions

- `uint addSubComponent` (const `SharedPtr< ModelComponent >` &subc)  
*Add a subcomponent to our internal list; return its index.*
- void `removeSubComponent` (const `SharedPtr< ModelComponent >` &subc)  
*Remove a subcomponent from our internal list, by address.*
- void `removeSubComponent` (const `uint` idx)  
*Remove a subcomponent from our internal list, by index.*
- void `removeSubComponent` (const `std::string` &tagname)  
*Remove a subcomponent from our internal list, by tagName.*
- void `removeAllSubComponents` ()  
*Remove all our subcomponents.*
- `SharedPtr< ModelComponent > subComponent` (const `uint` idx) const  
*Access a sub-component by index.*
- `SharedPtr< ModelComponent > subComponent` (const `std::string` &tagname) const  
*Access a sub-component by tagname.*
- `uint numSubComp` () const  
*Get current number of subcomponents.*
- bool `hasSubComponent` (const `std::string` &tagname, const bool recurse=false) const  
*Return true if we have a subcomponent by that tagname.*
- bool `hasSubComponent` (const `SharedPtr< ModelComponent >` &c, const bool recurse=false) const  
*Return true if we have the pointee as subcomponent.*

- void `printout` (std::ostream &s, const std::string &prefix="") const  
*Show our [ModelParam](#) internals and those of our subcomponents.*
- virtual void `reset` (const bool recurse=true)  
*Reset a model component and propagate the request to the sub-components.*

### Access to tunable ModelParam's

- virtual void `exportOptions` (const int whatToExport, const bool recurse=true)  
*Export some of our [ModelParam](#) parameters as command-line-options.*
- virtual bool `setModelParamString` (const std::string &name, const std::string &value, const bool recurse=false, const bool warn=true)  
*Parse a parameter value given as string and set the param.*
- template<class T> bool `setModelParamVal` (const std::string &name, const T &val, const bool recurse=false, const bool warn=true)  
*Set a parameter value.*
- std::string `getModelParamString` (const std::string &name, const bool recurse=false, const bool warn=true) const  
*Get a parameter value.*
- template<class T> bool `getModelParamVal` (const std::string &name, T &val, const bool recurse=false, const bool warn=true) const  
*Get a parameter value.*
- bool `doRequestOption` (const std::string &name, const bool useMyVal=false, const bool recurse=true, const bool warn=true)  
*External request that one of our [ModelParam](#) becomes an option.*

### ParamMap-based configuration functions

- void `readParamsFrom` (const ParamMap &pmap, const bool noerr=true)  
*Read params from the [ParamMap](#).*
- void `writeParamsTo` (ParamMap &pmap) const  
*Write params to the [ParamMap](#).*

## Protected Member Functions

- virtual void [start1](#) ()

*This is called from within [start\(\)](#) before the subcomponents start.*

- virtual void [start2](#) ()

*This is called from within [start\(\)](#) after the subcomponents have started.*

- virtual void [stop1](#) ()

*This is called from within [stop\(\)](#) before the subcomponents stop.*

- virtual void [stop2](#) ()

*This is called from within [stop\(\)](#) after the subcomponents have stopped.*

## Protected Attributes

- [ModelManager](#) \* **manager**

## Friends

- class [ModelParamBase](#)
- class [ModelManager](#)

### 10.164.1 Detailed Description

Base class for parameterized model components.

The goal is to provide a unified interface by which tunable parameters can be set, read, or exported as command-line options or in parameter maps or files, so that executables using a variable collection of model components can easily be built and configured using command-line options and config files. To this end, `ModelComponent` provides a set of facilities that allow it to hold a number of `ModelParam<T>` data members. A `ModelComponent` also holds a list of sub-components that are `ModelComponent` as well, and many of the `ModelComponent` functions will first apply to the component itself, then recursively to its subcomponents. The `ModelParam<T>` are the tunable parameters that are persistent and may be associated with a command-line option; see [ModelParam.H](#) for details. A `ModelComponent` is attached to a [ModelManager](#) that will handle option parsing and `ModelParam<T>` load/save. A `ModelComponent` may request its master to associate some of its `ModelParam<T>` data members with command-line options. A global list of all possible command-line option definitions for our entire code base is provided in [ModelOptionDefs.C](#). Command-line parsing will set the value of ALL `ModelParam<T>` objects in all `ModelComponent` objects

and their subcomponents that have requested the option to the value specified on the command line. So this mechanism should be used sparingly, for parameters that may be used across a variety of models and a variety of ModelComponent derivatives.

There is one trick to be aware of with respect to the basic operation and setup of ModelComponent. For speed reasons, we want our [ModelParam](#) parameters to be true data members, so that we can access them as efficiently as normal data members (rather than having to first look them up by name). On the other hand, the ModelComponent must also be able to produce a list of all its [ModelParam](#) members, so that they can be saved to a [ParamMap](#). To resolve this, ModelComponent both holds each [ModelParam](#) as a regular data member, and a list of pointers to all its [ModelParam](#) members. This list is automatically maintained when the [ModelParam](#) objects are constructed and destroyed, so that the user does not have to worry about it.

Definition at line 87 of file ModelComponent.H.

## 10.164.2 Constructor & Destructor Documentation

### 10.164.2.1 ModelComponent::ModelComponent ([ModelManager](#) & mgr, const [std::string](#) & descrName, const [std::string](#) & tagName)

Constructor.

Typically, in the constructor you should perform all initializations that are independent of the values of your [ModelParam](#) data members. Since the command-line usually has not been parsed yet at the time you construct your various ModelComponent objects, however, the [start\(\)](#) function below is provided to allow you to perform additional initializations after the command-line has been parsed and the [ModelParam](#) data members have been configured.

#### Parameters:

*mgr* our model manager; see [ModelManager.H](#)

*descrName* a human-readable descriptive name

*tagname* a computer-oriented unique name (used for storage in [ParamMap](#))

Definition at line 44 of file ModelComponent.C.

References [ModelManager::debugMode\(\)](#).

### 10.164.2.2 ModelComponent::~ModelComponent () [virtual]

Virtual destructor ensures proper destruction of derived classes.

Definition at line 53 of file ModelComponent.C.

References [ModelManager::debugMode\(\)](#), [ModelManager::destroyComponent\(\)](#), and [stop\(\)](#).

### 10.164.3 Member Function Documentation

#### 10.164.3.1 `uint ModelComponent::addSubComponent (const SharedPtr < ModelComponent > & sub)`

Add a subcomponent to our internal list; return its index.

Definition at line 180 of file `ModelComponent.C`.

References `uint`.

Referenced by `VisualCortex::addChannel()`, `Brain::addModule()`, `ComplexChannel::addSubChan()`, `BeobotEffectors::BeobotEffectors()`, `BeoSub::BeoSub()`, `BeoSubDepthSensor::BeoSubDepthSensor()`, `Beowulf::Beowulf()`, `BlueChannelSurprise::BlueChannelSurprise()`, `BlueYellowChannelSurprise::BlueYellowChannelSurprise()`, `CameraControl::CameraControl()`, `CarControl::CarControl()`, `SingleChannelBeoServer::check()`, `ColorBandChannelSurprise::ColorBandChannelSurprise()`, `ConvolveChannelSurprise::ConvolveChannelSurprise()`, `DirectionChannelSurprise::DirectionChannelSurprise()`, `DummyChannelSurprise::DummyChannelSurprise()`, `FlickerChannelSurprise::FlickerChannelSurprise()`, `GaborChannelSurprise::GaborChannelSurprise()`, `GpsDev::GpsDev()`, `GreenChannelSurprise::GreenChannelSurprise()`, `HMR3300::HMR3300()`, `IntensityChannelSurprise::IntensityChannelSurprise()`, `JunctionChannelSurprise::JunctionChannelSurprise()`, `lcd::lcd()`, `ModelEyeHeadController::ModelEyeHeadController()`, `ModelWrapperSaccadeController::ModelWrapperSaccadeController()`, `Monkey2SaccadeController::Monkey2SaccadeController()`, `MonkeyEyeHeadController::MonkeyEyeHeadController()`, `MonkeySaccadeController::MonkeySaccadeController()`, `PedestrianChannelSurprise::PedestrianChannelSurprise()`, `PsychoDisplay::PsychoDisplay()`, `RadioDecoder::RadioDecoder()`, `RedChannelSurprise::RedChannelSurprise()`, `RedGreenChannelSurprise::RedGreenChannelSurprise()`, `ModelManager::registerComponent()`, `WinnerTakeAllConfigurator::setModelParamString()`, `VisualCortexConfigurator::setModelParamString()`, `TaskRelevanceMapConfigurator::setModelParamString()`, `SimulationViewerConfigurator::setModelParamString()`, `SaliencyMapConfigurator::setModelParamString()`, `SaccadeControllerConfigurator::setModelParamString()`, `RetinaConfigurator::setModelParamString()`, `InferoTemporalConfigurator::setModelParamString()`, `GistEstimatorConfigurator::setModelParamString()`, `FrameGrabberConfigurator::setModelParamString()`, `AttentionGuidanceMapConfigurator::setModelParamString()`, `SimpleEyeHeadSaccadeController::SimpleEyeHeadSaccadeController()`, `SimulationViewerEyeMvt2::SimulationViewerEyeMvt2()`, `SingleChannelBeoServer::SingleChannelBeoServer()`, `SOBlueYellowChannelSurprise::SOBlueYellowChannelSurprise()`, `SOGreenRedChannelSurprise::SOGreenRedChannelSurprise()`, `SORedGreenChannelSurprise::SORedGreenChannelSurprise()`, `SOYellowBlueChannelSurprise::SOYellowBlueChannelSurprise()`, `SSC::SSC()`, `VisualCortexEyeMvt::start1()`, `StdBrain::StdBrain()`, `TCPcommunicator::TCPcommunicator()`, `TemplateMatchChannelSurprise::TemplateMatchChannelSurprise()`, `ThresholdFrictionSaccadeController::ThresholdFrictionSaccadeController()`, `VCC4::VCC4()`,



VisualCortexEyeMvt::VisualCortexEyeMvt(), and YellowChannelSurprise::YellowChannelSurprise().

#### 10.164.3.2 `std::string ModelComponent::descriptiveName () const`

Get a human-readable descriptive name for this component.

This is NOT intended to be a name for programmatic use (such as for indexing into associative arrays). It is not necessarily unique on a per-object basis; it's possible that all objects of the same class return the same `descriptiveName()`.

Definition at line 164 of file ModelComponent.C.

Referenced by `ComplexChannel::combineOutputs()`, `SingleChannel::combineSubMaps()`, `ImageDisplay::display()`, `SingleChannel::getSubmapName()`, `DirectFeedChannel::getSubmapName()`, `SingleChannelBeo::requestBeowulfNode()`, `SingleChannelBeo::sendInput()`, and `FrameSeries::update()`.

#### 10.164.3.3 `void ModelComponent::destroyManager ()`

Signify that our `ModelManager` just got destroyed.

This should not be called manually, but will be called from within the `ModelManager` destructor. It will ensure that we do not try to unregister with our `ModelManager` if the manager is destroyed before us.

Definition at line 116 of file ModelComponent.C.

Referenced by `ModelManager::~~ModelManager()`.

#### 10.164.3.4 `bool ModelComponent::doRequestOption (const std::string & name, const bool useMyVal = false, const bool recurse = true, const bool warn = true)`

External request that one of our `ModelParam` becomes an option.

This is to allow external callers to selectively decide that some of our internal `ModelParam` parameters should be exported as command-line options. If `recurse` is true, will look for the `ModelParam` in all our subcomponents, and will link all matching `ModelParam` in us and our subcomponents to the option (so that if a command-line value is provided, it will affect all matching `ModelParam` objects in us and all our subcomponents). Will return false (and generate a warning message if `warn` is true) if we (or our subcomponents if `recurse` is true) don't have a `ModelParam` of the specified name. Will generate a fatal error if we have the `ModelParam` but no known command-line option exists for that name in `ModelOptionDefs.C`. Returns true if the `ModelParam` was found and added as a command-line option. Throws a fatal error if we are started.

Definition at line 367 of file ModelComponent.C.

References `ModelManager::requestOption()`.

Referenced by `RadioDecoder::exportOptions()`.

#### 10.164.3.5 `void ModelComponent::exportOptions (const int whatToExport, const bool recurse = true) [virtual]`

Export some of our [ModelParam](#) parameters as command-line-options.

This should be called once on the model manager to decide which sets of [ModelParam](#) parameters should become command-line options; the call will then recurse through all `ModelComponent` objects that have been registered with the manager. This is provided so that external meta-knowledge about what a collection of `ModelComponent` objects forming a computational model can be used to decide on which command-line options are relevant: for instance, if you build a model that has no [OutputFrameSeries](#), then you will not be able to call `saveResults()` on your [Brain](#) or other objects, so you would not want to export command-line options that relate to which results should be saved. See [ModelOptionDefs.H](#) for the various classes of options that you may export. PROGRAMMER NOTE: when you overload this in one of your derived classes, be sure to call the base function somewhere in your overload; the base function just implements the recursion through the subcomponents, and your overload should implement a series of calls to [ModelManager::requestOption\(\)](#) depending on the contents of the `whatToExport` argument.

Reimplemented in [AttentionGuidanceMap](#), [AttentionGuidanceMapConfigurator](#), [AudioGrabber](#), [AudioMixer](#), [Beowulf](#), [Brain](#), [ChannelBase](#), [GaborChannel](#), [DirectionChannel](#), [JunctionChannel](#), [TcorrChannel](#), [ScorrChannel](#), [DummyChannel](#), [DisparityChannel](#), [ColorChannel](#), [OrientationChannel](#), [MultiColorBandChannel](#), [MotionChannel](#), [StereoChannel](#), [LJunctionChannel](#), [TJunctionChannel](#), [XJunctionChannel](#), [EndPointChannel](#), [ComplexChannel](#), [DirectFeedChannel](#), [FrameGrabber](#), [FrameGrabberConfigurator](#), [FrameSeries](#), [GistEstimator](#), [GistEstimatorConfigurator](#), [ImageDisplay](#), [InferoTemporal](#), [InferoTemporalConfigurator](#), [MbariResultViewer](#), [ModelManager](#), [MPEGStream](#), [PsychoDisplay](#), [RadioDecoder](#), [RetinaConfigurator](#), [RetinaStd](#), [SaccadeControllerConfigurator](#), [TrivialSaccadeController](#), [FixedSaccadeController](#), [FrictionSaccadeController](#), [ThresholdSaccadeController](#), [Threshold-FrictionSaccadeController](#), [SimpleEyeHeadSaccadeController](#), [MonkeyEyeHeadController](#), [Monkey2SaccadeController](#), [MonkeySaccadeController](#), [EyeTrackerSaccadeController](#), [VisBufSaccadeController](#), [ModelEyeHeadController](#), [ModelWrapperSaccadeController](#), [SaliencyMap](#), [SaliencyMapConfigurator](#), [SaliencyMapStd](#), [SaliencyMapFast](#), [ShapeEstimator](#), [SimulationViewerConfigurator](#), [SimulationViewerCompress](#), [SimulationViewerEyeMvt](#), [SimulationViewerEyeMvt2](#), [SimulationViewerStd](#), [SingleChannel](#), [SingleChannelBeoServer](#), [SingleChannelSurprise](#), [SockServ](#), [SoxChannel](#), [TaskRelevanceMap](#), [TaskRelevanceMapConfigurator](#), [TaskRelevanceMapStd](#), [TaskRelevanceMapKillStatic](#), [TaskRelevanceMapKillN](#), [TCPcommunicator](#), [VCC4](#), [VisualBuffer](#), [VisualCortex](#), [VisualCortexConfigurator](#), [VisualCortexEyeMvt](#), [WinnerTakeAll](#), and [WinnerTakeAllConfigurator](#).

Definition at line 287 of file ModelComponent.C.

References ModelManager::debugMode().

Referenced by SoxChannel::buildSubChans(), MotionChannelSurprise::buildSubChans(), MultiColorBandChannelSurprise::buildSubChans(), MotionChannelBeo::buildSubChans(), MultiColorBandChannelBeo::buildSubChans(), StereoChannel::buildSubChans(), MotionChannel::buildSubChans(), MultiColorBandChannel::buildSubChans(), WinnerTakeAllConfigurator::exportOptions(), WinnerTakeAll::exportOptions(), VisualCortexConfigurator::exportOptions(), VisualCortex::exportOptions(), VisualBuffer::exportOptions(), VCC4::exportOptions(), TCPcommunicator::exportOptions(), TaskRelevanceMapConfigurator::exportOptions(), TaskRelevanceMap::exportOptions(), SockServ::exportOptions(), SingleChannelSurprise::exportOptions(), SingleChannelBeoServer::exportOptions(), SimulationViewerStd::exportOptions(), SimulationViewerEyeMvt2::exportOptions(), SimulationViewerEyeMvt::exportOptions(), SimulationViewerCompress::exportOptions(), SimulationViewerConfigurator::exportOptions(), ShapeEstimator::exportOptions(), SaliencyMapFast::exportOptions(), SaliencyMapConfigurator::exportOptions(), SaliencyMap::exportOptions(), ModelEyeHeadController::exportOptions(), ModelWrapperSaccadeController::exportOptions(), VisBufSaccadeController::exportOptions(), EyeTrackerSaccadeController::exportOptions(), MonkeySaccadeController::exportOptions(), Monkey2SaccadeController::exportOptions(), MonkeyEyeHeadController::exportOptions(), SimpleEyeHeadSaccadeController::exportOptions(), ThresholdFrictionSaccadeController::exportOptions(), ThresholdSaccadeController::exportOptions(), FrictionSaccadeController::exportOptions(), FixedSaccadeController::exportOptions(), TrivialSaccadeController::exportOptions(), SaccadeControllerConfigurator::exportOptions(), RetinaStd::exportOptions(), RetinaConfigurator::exportOptions(), PsychoDisplay::exportOptions(), MPEGStream::exportOptions(), ModelManager::exportOptions(), MbariResultViewer::exportOptions(), InferoTemporalConfigurator::exportOptions(), InferoTemporal::exportOptions(), ImageDisplay::exportOptions(), GistEstimatorConfigurator::exportOptions(), FrameSeries::exportOptions(), FrameGrabberConfigurator::exportOptions(), FrameGrabber::exportOptions(), ChannelBase::exportOptions(), Brain::exportOptions(), Beowulf::exportOptions(), AudioMixer::exportOptions(), AudioGrabber::exportOptions(), AttentionGuidanceMapConfigurator::exportOptions(), and AttentionGuidanceMap::exportOptions().

**10.164.3.6** `std::string ModelComponent::getModelParamString (const std::string & name, const bool recurse = false, const bool warn = true) const`

Get a parameter value.

Because the parameter may exist both in us and in several of our subcomponents, we will stop and return the value of the first instance encountered, searching first in us, then recursively in our subcomponents (depth-first search, if recurse is true), in the

order in which they were added. If the parameter is not found in us or any of our subcomponents, the string "<unknown>" is returned (and a warning is printed out, if warn is true). It is ok to call this on a started ModelComponent.

Definition at line 339 of file ModelComponent.C.

**10.164.3.7** `template<class T> bool ModelComponent::getModelParamVal  
(const std::string & name, T & val, const bool recurse = false, const  
bool warn = true) const [inline]`

Get a parameter value.

This is slow, as it will search for the [ModelParam](#) by name. If you need efficient access to [ModelParam](#) values, you should define a wrapper method in your ModelComponent derivative. In normal operation, indeed, only the methods of the ModelComponent derivative should directly access the [ModelParam](#), not external people. Because the parameter may exist both in us and in several of our subcomponents, we will stop and return the value of the first instance encountered, searching first in us, then recursively in our subcomponents (depth-first search, if recurse is true), in the order in which they were added. If the parameter is not found in us or any of our subcomponents, false is returned and val is unchanged (and a warning is issued if warn is true). It is ok to call this on a started ModelComponent.

Definition at line 482 of file ModelComponent.H.

Referenced by JunctionChannel::doInput(), SimulationViewerEyeMvt::eye-Movement(), SimulationViewerCompress::getTraj(), VisualCortexEyeMvt::start1(), and InputFrameSeries::start2().

**10.164.3.8** `bool ModelComponent::hasSubComponent (const SharedPtr<  
ModelComponent > & c, const bool recurse = false) const`

Return true if we have the pointee as subcomponent.

**Parameters:**

*recurse* if true, will look not only in our subcomponents but also recursively in their subcomponents (depth-first search)

Definition at line 269 of file ModelComponent.C.

References SharedPtr< T >::get().

**10.164.3.9** `bool ModelComponent::hasSubComponent (const std::string &  
tagname, const bool recurse = false) const`

Return true if we have a subcomponent by that tagname.

**Parameters:**

***recurse*** if true, will look not only in our subcomponents but also recursively in their subcomponents (depth-first search)

Definition at line 251 of file ModelComponent.C.

Referenced by VisualCortex::hasChannel(), MbariResultViewer::MbariResultViewer(), and Brain::removeModule().

**10.164.3.10 [uint](#) ModelComponent::numSubComp () const**

Get current number of subcomponents.

Definition at line 247 of file ModelComponent.C.

References [uint](#).

Referenced by VisualCortex::numChannels(), ComplexChannel::numChans(), and SimulationViewerCompress::start1().

**10.164.3.11 void ModelComponent::printout (std::ostream & s, const [std::string](#) & *prefix* = "") const**

Show our [ModelParam](#) internals and those of our subcomponents.

Definition at line 128 of file ModelComponent.C.

Referenced by ModelManager::start1().

**10.164.3.12 void ModelComponent::readParamsFrom (const [ParamMap](#) & *pmap*, const bool *noerr* = true)**

Read params from the [ParamMap](#).

Throws a fatal error if we are started.

**Parameters:**

***noerr*** will not generate an error message if some of our parameters do not exist in [ParamMap](#)

Definition at line 397 of file ModelComponent.C.

References [ParamMap::getSubmap\(\)](#), and [ParamMap::hasParam\(\)](#).

Referenced by ModelManager::loadConfig(), and ModelManager::parseCommandLine().

**10.164.3.13 void ModelComponent::removeAllSubComponents ()**

Remove all our subcomponents.

Definition at line 220 of file ModelComponent.C.

Referenced by SoxChannel::buildSubChans(), EndPointChannelSurprise::buildSubChans(), XJunctionChannelSurprise::buildSubChans(), TJunctionChannelSurprise::buildSubChans(), LJunctionChannelSurprise::buildSubChans(), MotionChannelSurprise::buildSubChans(), MultiColorBandChannelSurprise::buildSubChans(), OrientationChannelSurprise::buildSubChans(), MotionChannelBeo::buildSubChans(), MultiColorBandChannelBeo::buildSubChans(), OrientationChannelBeo::buildSubChans(), StereoChannel::buildSubChans(), EndPointChannel::buildSubChans(), XJunctionChannel::buildSubChans(), TJunctionChannel::buildSubChans(), LJunctionChannel::buildSubChans(), MotionChannel::buildSubChans(), MultiColorBandChannel::buildSubChans(), OrientationChannel::buildSubChans(), SingleChannelBeoServer::check(), ColorChannelSurprise::ColorChannelSurprise(), MultiConvolveChannelSurprise::MultiConvolveChannelSurprise(), and SOColorChannelSurprise::SOColorChannelSurprise().

**10.164.3.14 void ModelComponent::removeSubComponent (const [std::string](#) & tagname)**

Remove a subcomponent from our internal list, by tagName.

This will only remove the component from our list of registered subcomponents, but will not attempt to destroy it.

Definition at line 208 of file ModelComponent.C.

**10.164.3.15 void ModelComponent::removeSubComponent (const [uint](#) idx)**

Remove a subcomponent from our internal list, by index.

This will only remove the component from our list of registered subcomponents, but will not attempt to destroy it.

Definition at line 197 of file ModelComponent.C.

References uint.

**10.164.3.16 void ModelComponent::removeSubComponent (const [SharedPtr](#)<[ModelComponent](#) > & subc)**

Remove a subcomponent from our internal list, by address.

This will only remove the component from our list of registered subcomponents, but will not attempt to destroy it.

Definition at line 184 of file ModelComponent.C.

References `SharedPtr< T >::get()`.

Referenced by `SingleChannelBeoServer::check()`, `Brain::removeModule()`, `WinnerTakeAllConfigurator::setModelParamString()`, `VisualCortexConfigurator::setModelParamString()`, `TaskRelevanceMapConfigurator::setModelParamString()`, `SimulationViewerConfigurator::setModelParamString()`, `SaliencyMapConfigurator::setModelParamString()`, `SaccadeControllerConfigurator::setModelParamString()`, `RetinaConfigurator::setModelParamString()`, `InferoTemporalConfigurator::setModelParamString()`, `GistEstimatorConfigurator::setModelParamString()`, `FrameGrabberConfigurator::setModelParamString()`, `AttentionGuidanceMapConfigurator::setModelParamString()`, `MultiDirectFeedChannel::setNumChans()`, and `ModelManager::unregisterComponent()`.

#### 10.164.3.17 `void ModelComponent::reset (const bool recurse = true)` [virtual]

Reset a model component and propagate the request to the sub-components.

For each ModelComponent reset should set it into the state that it was just after calling the constructor. It will depend on the component what exactly that involves. PROGRAMMER NOTE: when you overload this in one of your derived classes, be sure to call the base function somewhere in your overload; the base function just implements the recursion through the subcomponents

Reimplemented in [Brain](#), [FlickerChannel](#), [ComplexChannel](#), [DirectFeedChannel](#), [FrameSeries](#), [MbariResultViewer](#), [SaccadeController](#), [ThresholdSaccadeController](#), [ShapeEstimator](#), [SimulationViewerStd](#), [SingleChannel](#), [Staircase](#), [VisualCortex](#), and [VisualCortexEyeMvt](#).

Definition at line 481 of file ModelComponent.C.

Referenced by `VisualCortex::reset()`, `Staircase::reset()`, `SingleChannel::reset()`, `SimulationViewerStd::reset()`, `ShapeEstimator::reset()`, `SaccadeController::reset()`, `MbariResultViewer::reset()`, `FrameSeries::reset()`, `DirectFeedChannel::reset()`, `ComplexChannel::reset()`, `Brain::reset()`, and `BeoSubMotor::~BeoSubMotor()`.

#### 10.164.3.18 `void ModelComponent::setDescriptiveName (const std::string & name)`

Change this component's descriptive name.

Definition at line 168 of file ModelComponent.C.

Referenced by `ColorBandChannel::ColorBandChannel()`, `DirectFeedChannel::DirectFeedChannel()`, `DirectionChannel::DirectionChannel()`, `DisparityChannel::DisparityChannel()`, `GaborChannel::GaborChannel()`, and `JunctionChannel::Junction-`

Channel().

**10.164.3.19** `bool ModelComponent::setModelParamString (const std::string & name, const std::string & value, const bool recurse = false, const bool warn = true) [virtual]`

Parse a parameter value given as string and set the param.

Should not be needed in normal operation, inefficient text-based interface. In normal operation, the internals of the ModelComponent derivative will directly access the [ModelParam](#), not external people. If you want to provide external access to your [ModelParam](#) values, you should implement a wrapper for them in your ModelComponent derivative (e.g., implement a function setSize() that will set the value of your Size [ModelParam](#)). Returns true if the [ModelParam](#) was found and set. Throws a fatal error if the ModelComponent was started, to avoid external people messing around with our [ModelParam](#) values during operation of the component. PROGRAMMER NOTE: This is called by the [ModelManager](#) to update us whenever one of our [ModelParam](#) members for which we have requested a command-line option is being changed, and it is also called on each [ModelParam](#) when we do a [readParamsFrom\(\)](#); so, you may want to overload this call if you need to do any internal reconfiguration as some of your parameters change; see [OrientationChannel](#) in [Channels.C](#) for an example; also see [SaccadeControllerConfigurator](#) in [SaccadeControllers.H](#) for how to use this to select subcomponents at runtime.

**Parameters:**

**name** the [ModelParam](#) tagname

**value** the parameter value, to be parsed

**recurse** will propagate this call to all our subcomponents and their subs as well

**warn** will produce a warning message if parameter not found

Reimplemented in [AttentionGuidanceMapConfigurator](#), [Beowulf](#), [OrientationChannel](#), [MultiColorBandChannel](#), [MotionChannel](#), [StereoChannel](#), [LJunctionChannel](#), [TJunctionChannel](#), [XJunctionChannel](#), [EndPointChannel](#), [FrameGrabberConfigurator](#), [GistEstimatorConfigurator](#), [InferoTemporalConfigurator](#), [MbariResultViewer](#), [ModelManager](#), [RetinaConfigurator](#), [SaccadeControllerConfigurator](#), [SaliencyMapConfigurator](#), [SimulationViewerConfigurator](#), [SimulationViewerCompress](#), [SoxChannel](#), [TaskRelevanceMapConfigurator](#), [VisualCortexConfigurator](#), and [WinnerTakeAllConfigurator](#).

Definition at line 306 of file ModelComponent.C.

Referenced by [ModelParamBase::readFrom\(\)](#), [ModelManager::requestOption\(\)](#), [WinnerTakeAllConfigurator::setModelParamString\(\)](#), [VisualCortexConfigurator::setModelParamString\(\)](#), [TaskRelevanceMapConfigurator::setModelParamString\(\)](#),



SoxChannel::setModelParamString(), SimulationViewerCompress::setModelParamString(), SimulationViewerConfigurator::setModelParamString(), SaliencyMapConfigurator::setModelParamString(), SaccadeControllerConfigurator::setModelParamString(), RetinaConfigurator::setModelParamString(), ModelManager::setModelParamString(), MbariResultViewer::setModelParamString(), InferoTemporalConfigurator::setModelParamString(), GistEstimatorConfigurator::setModelParamString(), FrameGrabberConfigurator::setModelParamString(), StereoChannel::setModelParamString(), EndPointChannel::setModelParamString(), XJunctionChannel::setModelParamString(), TJunctionChannel::setModelParamString(), LJunctionChannel::setModelParamString(), MotionChannel::setModelParamString(), MultiColorBandChannel::setModelParamString(), OrientationChannel::setModelParamString(), Beowulf::setModelParamString(), and AttentionGuidanceMapConfigurator::setModelParamString().

**10.164.3.20** `template<class T> bool ModelComponent::setModelParamVal  
(const std::string & name, const T & val, const bool recurse = false,  
const bool warn = true) [inline]`

Set a parameter value.

This is slow, as it will search for the [ModelParam](#) by name. If you need efficient access to [ModelParam](#) values, you should define a wrapper method in your ModelComponent derivative. In normal operation, indeed, only the methods of the ModelComponent derivative should directly access the [ModelParam](#), not external people. Returns true if the [ModelParam](#) was found and set. Throws a fatal error if the ModelComponent was started, to avoid external people messing around with our [ModelParam](#) values during operation of the component.

#### Parameters:

*name* the [ModelParam](#) tagname

*val* the parameter value

*recurse* will propagate this call to all our subcomponents and their subs as well

*warn* will produce a warning message if parameter not found

Definition at line 442 of file ModelComponent.H.

Referenced by InputFrameSeries::start2(), and AudioGrabber::start2().

**10.164.3.21** `void ModelComponent::setTagName (const std::string & name)`

Change this component's tag name.

Definition at line 176 of file ModelComponent.C.

Referenced by `ColorBandChannel::ColorBandChannel()`, `DirectFeedChannel::DirectFeedChannel()`, `DirectionChannel::DirectionChannel()`, `DisparityChannel::DisparityChannel()`, `GaborChannel::GaborChannel()`, and `JunctionChannel::JunctionChannel()`.

#### 10.164.3.22 `void ModelComponent::start ()`

Start the component.

This will start the component by triggering the following sequence of actions:

- Check if we are already started and emit a fatal error if so
- Call `start1()` that may be overloaded by derived classes
- Call `start()` on all our subcomponents
- Call `start2()` that may be overloaded by derived classes
- Switch us to "started" state, in which modifying our `ModelParam` values externally is forbidden

Derived classes should overload `start1()` and `start2()` as a way to get the component started, once the command-line options and config files have been parsed and all the `ModelParam` objects have been configured. Default no-op implementations are provided for `start1()` and `start2()`, so if you don't need to do anything to start your component, you don't need to implement `start1()` and `start2()`. Because this is a recursive call on our sub-components, typically you only need to call it once on your `ModelManager` to get your entire hierarchy of `ModelComponent` objects started.

Definition at line 68 of file `ModelComponent.C`.

References `ModelManager::debugMode()`, `start1()`, and `start2()`.

Referenced by `main()`, and `V4LgrabberLight::V4LgrabberLight()`.

#### 10.164.3.23 `void ModelComponent::start1 ()` [protected, virtual]

This is called from within `start()` before the subcomponents start.

Reimplemented in `BeoChip`, `Brain`, `GaborChannel`, `TemplateMatchChannel`, `PedestrianChannel`, `DisparityChannel`, `MultiConvolveChannel`, `StereoChannel`, `IntensityChannelBeo`, `RedGreenChannelBeo`, `BlueYellowChannelBeo`, `GaborChannelBeo`, `FlickerChannelBeo`, `ColorBandChannelBeo`, `DirectionChannelBeo`, `COMPort`, `IEEE1394grabber`, `InferoTemporal`, `ModelManager`, `RadioDecoder`, `FrictionSaccadeController`, `ThresholdSaccadeController`, `EyeTrackerSaccadeController`, `VisBufSaccadeController`, `Serial`, `SimulationViewerCompress`, `SimulationViewerEyeMvt`, `SimulationViewerEyeMvt2`, `SingleChannel`, `SingleChannelSurprise`,

[StdBrain](#), [TaskRelevanceMapKillStatic](#), [TCPcommunicator](#), [V4Lgrabber](#), [VCC4](#), [VisualBuffer](#), and [VisualCortexEyeMvt](#).

Definition at line 148 of file ModelComponent.C.

Referenced by [start\(\)](#).

#### 10.164.3.24 void ModelComponent::start2 () [protected, virtual]

This is called from within [start\(\)](#) after the subcomponents have started.

Reimplemented in [AudioGrabber](#), [AudioMixer](#), [BeoSubDepthSensor](#), [Complex-Channel](#), [FrameSeries](#), [InputFrameSeries](#), [HMR3300](#), [PsychoDisplay](#), [RadioDecoder](#), [RetinaStd](#), [Staircase](#), and [TCPcommunicator](#).

Definition at line 152 of file ModelComponent.C.

Referenced by [start\(\)](#).

#### 10.164.3.25 bool ModelComponent::started () const

Did we already get a call to [start\(\)](#)?

Definition at line 90 of file ModelComponent.C.

Referenced by [TCPcommunicator::contact\(\)](#), [InputFrameSeries::getImageDims\(\)](#), [SingleChannel::killCaches\(\)](#), [Beowulf::requestNode\(\)](#), and [FrameSeries::setFileStem\(\)](#).

#### 10.164.3.26 void ModelComponent::stop ()

Stop the component.

This will stop the component by triggering the following sequence of actions:

- Check if we are already started and emit a fatal error if not
- Call [stop1\(\)](#) that may be overloaded by derived classes
- Call [stop\(\)](#) on all our subcomponents
- Call [stop2\(\)](#) that may be overloaded by derived classes
- Switch us out of "started" state, so that modifying our [ModelParam](#) values externally is allowed

Derived classes should overload [stop1\(\)](#) and [stop2\(\)](#) as a way to get the component stopped, once the command-line options and config files have been parsed and all the [ModelParam](#) and objects have been configured. Default no-op implementations are

provided for [stop1\(\)](#) and [stop2\(\)](#), so if you don't need to do anything to stop your component, you don't need to implement [stop1\(\)](#) and [stop2\(\)](#). Because this is a recursive call on our sub-component, typically you only need to call it once on your [ModelManager](#) to get the entire hierarchy of ModelComponent objects stopped.

Definition at line 94 of file ModelComponent.C.

References [ModelManager::debugMode\(\)](#), [stop1\(\)](#), and [stop2\(\)](#).

Referenced by [main\(\)](#), [~ModelComponent\(\)](#), and [V4LgrabberLight::~~V4LgrabberLight\(\)](#).

#### 10.164.3.27 **void ModelComponent::stop1 ()** [protected, virtual]

This is called from within [stop\(\)](#) before the subcomponents stop.

Reimplemented in [AudioGrabber](#), [AudioMixer](#), [BeoSubDepthSensor](#), [HMR3300](#), [PsychoDisplay](#), [RadioDecoder](#), [EyeTrackerSaccadeController](#), [VisBufSaccadeController](#), [SimulationViewerCompress](#), [SimulationViewerEyeMvt](#), [SimulationViewerEyeMvt2](#), [Staircase](#), [TCPcommunicator](#), and [VCC4](#).

Definition at line 156 of file ModelComponent.C.

Referenced by [stop\(\)](#).

#### 10.164.3.28 **void ModelComponent::stop2 ()** [protected, virtual]

This is called from within [stop\(\)](#) after the subcomponents have stopped.

Reimplemented in [BeoChip](#), [IEEE1394grabber](#), [ModelManager](#), [MPEGStream](#), [SerialSingleChannel](#), and [V4Lgrabber](#).

Definition at line 160 of file ModelComponent.C.

Referenced by [stop\(\)](#).

#### 10.164.3.29 **SharedPtr< ModelComponent > ModelComponent::sub-Component (const std::string & tagname) const**

Access a sub-component by tagname.

Definition at line 234 of file ModelComponent.C.

### 10.164.3.30 [SharedPtr](#)< [ModelComponent](#) > [ModelComponent::subComponent](#) (const [uint](#) *idx*) const

Access a sub-component by index.

Definition at line 224 of file ModelComponent.C.

References [uint](#).

Referenced by [VisualCortex::channel\(\)](#), [CompositeColorChannel::channel\(\)](#), [VisualCortex::channelPtr\(\)](#), [VisualCortexEyeMvt::start1\(\)](#), [SimulationViewer-Compress::start1\(\)](#), and [ComplexChannel::subChan\(\)](#).

### 10.164.3.31 [std::string](#) [ModelComponent::tagName](#) () const

Get a programmer-usable tag name for this object.

This is intended to be unique among component objects, so that this string could be used for associate-array indexing, for example.

Definition at line 172 of file ModelComponent.C.

Referenced by [ComplexChannel::combineOutputs\(\)](#), [SingleChannel::combineSubMaps\(\)](#), [HueChannel::combineSubMaps\(\)](#), [FrameSeries::computeFileName\(\)](#), [ImageDisplay::display\(\)](#), [VisualCortex::findLikelihood\(\)](#), [ComplexChannel::findLikelihood\(\)](#), [VisualCortex::getOutput\(\)](#), [SingleChannel::getRawCSmap\(\)](#), [VisualCortex::input\(\)](#), [SingleChannel::learn\(\)](#), [SingleChannel::postProcessMap\(\)](#), [VisualCortex::readFrom\(\)](#), [ComplexChannel::readFrom\(\)](#), [VisualCortex::reco\(\)](#), [ComplexChannel::reco\(\)](#), [SingleChannelBeo::requestBeowulfNode\(\)](#), [SingleChannel::saveResults\(\)](#), [ComplexChannel::saveResults\(\)](#), [SingleChannelBeo::sendInput\(\)](#), [VisualCortex::writeTo\(\)](#), and [ComplexChannel::writeTo\(\)](#).

### 10.164.3.32 [void](#) [ModelComponent::writeParamsTo](#) ([ParamMap](#) & *pmap*) const

Write params to the [ParamMap](#).

Prints out a warning message if we are started.

Definition at line 419 of file ModelComponent.C.

References [ParamMap::putSubmap\(\)](#).

Referenced by [ModelManager::saveConfig\(\)](#), and [SingleChannelBeo::sendInput\(\)](#).

The documentation for this class was generated from the following files:

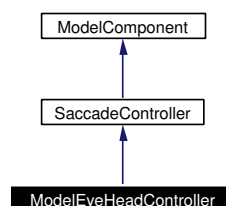
- [ModelComponent.H](#)
- [ModelComponent.C](#)

## 10.165 ModelEyeHeadController Class Reference

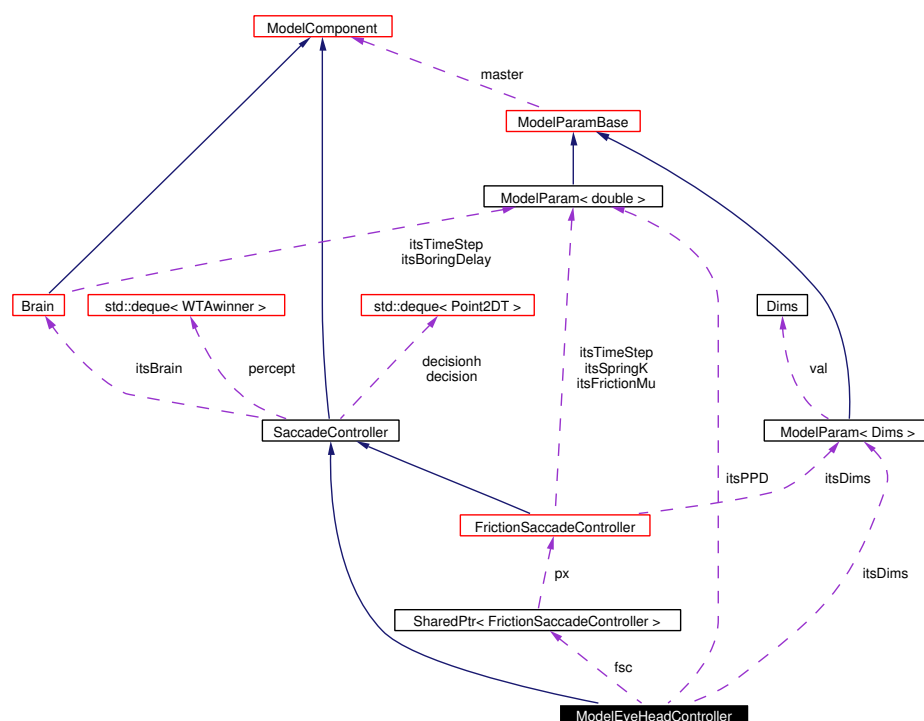
Realistic Eye/Head controller.

```
#include <SaccadeControllers.H>
```

Inheritance diagram for ModelEyeHeadController:



Collaboration diagram for ModelEyeHeadController:



## Public Member Functions

- [ModelEyeHeadController](#) ([ModelManager](#) &mgr, const [std::string](#) paramprefix="SCeye")  
*Constructor.*
- [~ModelEyeHeadController](#) ()  
*Destructor.*
- virtual void [exportOptions](#) (const int whatToExport, const bool recurse=true)  
*Export some of our [ModelParam](#) parameters as command-line-options.*
- void [saccade](#) (const [Point2DT](#) &target)  
*Initiate a realistic saccade.*
- void [pix2deg](#) (const [Point2D](#) &pixloc, double &xdeg, double &ydeg) const  
*Helper function converting from pixels to degrees.*
- void [deg2pix](#) (const double xdeg, const double ydeg, [Point2D](#) &pixloc) const  
*Helper function converting from degrees to pixels.*
- virtual void [evolve](#) (const double t)  
*evolve one time step (called, e.g., by [Brain::evolve\(\)](#))*

## Protected Member Functions

- virtual void [computeWhenNewPercept](#) ()  
*This method is called each time a new percept has arrived.*
- virtual void [computeWhenResetPos](#) ()  
*This method is called each time a reset is made.*
- virtual [Point2D](#) [computeWhenNewDecision](#) (const double t)  
*This method is called each time a new decision is requested.*

## Protected Attributes

- [ModelParam](#)< double > itsPPD  
*pixels per degree*

- [ModelParam](#) < [Dims](#) > [itsDims](#)

*Image size.*

### 10.165.1 Detailed Description

Realistic Eye/Head controller.

This is part of an attempt at a simplified approach to the [MonkeySaccadeController](#), which turned out quite complex because of the interdependency between head and eye movements. Here, we fake it by having independent controllers for the eyes and the head, such that we don't need to even deal with gaze stabilization during head movements and VOR. This controller will do either the head or the eye. For a controller that does both, see [Monkey2SaccadeController](#).

Definition at line 787 of file [SaccadeControllers.H](#).

### 10.165.2 Constructor & Destructor Documentation

#### 10.165.2.1 [ModelEyeHeadController::ModelEyeHeadController](#) ([ModelManager](#) & *mgr*, const [std::string](#) *paramprefix* = "SCeye")

Constructor.

##### Parameters:

*paramprefix* As this controller may be used either for the eye or for the head (as part of bigger controllers that include both eye and head), this should be either "SCeye" or "SChead" and will determine which model options we use for our internal parameters (eye or head)

Definition at line 2261 of file [SaccadeControllers.C](#).

References [ModelComponent::addSubComponent\(\)](#).

#### 10.165.2.2 [ModelEyeHeadController::~~ModelEyeHeadController](#) ()

Destructor.

Definition at line 2276 of file [SaccadeControllers.C](#).



### 10.165.3 Member Function Documentation

#### 10.165.3.1 **Point2D** ModelEyeHeadController::computeWhenNewDecision (const double t) [protected, virtual]

This method is called each time a new decision is requested.

Implements [SaccadeController](#).

Definition at line 2443 of file SaccadeControllers.C.

References [SaccadeController::getPreviousDecision\(\)](#), [Point2D::i](#), and [Point2DT::p](#).

#### 10.165.3.2 void ModelEyeHeadController::computeWhenNewPercept () [protected, virtual]

This method is called each time a new percept has arrived.

Implements [SaccadeController](#).

Definition at line 2366 of file SaccadeControllers.C.

References [SaccadeController::getPreviousPercept\(\)](#).

#### 10.165.3.3 void ModelEyeHeadController::computeWhenResetPos () [protected, virtual]

This method is called each time a reset is made.

Implements [SaccadeController](#).

Definition at line 2428 of file SaccadeControllers.C.

References [SaccadeController::endSaccade\(\)](#), [SaccadeController::getPreviousPercept\(\)](#), [SaccadeController::isInSaccade\(\)](#), [WTAwinner::p](#), [pix2deg\(\)](#), and [WTAwinner::t](#).

#### 10.165.3.4 void ModelEyeHeadController::deg2pix (const double xdeg, const double ydeg, **Point2D** & pixloc) const

Helper function converting from degrees to pixels.

Definition at line 2352 of file SaccadeControllers.C.

Referenced by [evolve\(\)](#).

**10.165.3.5 void ModelEyeHeadController::evolve (const double t)  
[virtual]**

evolve one time step (called, e.g., by [Brain::evolve\(\)](#))

Implements [SaccadeController](#).

Definition at line 2378 of file SaccadeControllers.C.

References [deg2pix\(\)](#), [Point2D::i](#), [SaccadeController::isInSaccade\(\)](#), and [Point2D::j](#).

**10.165.3.6 void ModelEyeHeadController::exportOptions (const int  
*whatToExport*, const bool *recurse* = true) [virtual]**

Export some of our [ModelParam](#) parameters as command-line-options.

See the base function in [ModelComponent.H](#) for info.

Reimplemented from [ModelComponent](#).

Definition at line 2280 of file SaccadeControllers.C.

References [ModelComponent::exportOptions\(\)](#), [OPTEXP\\_CORE](#), and [ModelManager::requestOption\(\)](#).

**10.165.3.7 void ModelEyeHeadController::pix2deg (const [Point2D](#) & *piloc*,  
double & *xdeg*, double & *ydeg*) const**

Helper function converting from pixels to degrees.

Definition at line 2338 of file SaccadeControllers.C.

References [ModelParam< double >::getVal\(\)](#), [ModelParam< Dims >::getVal\(\)](#), [Dims::h\(\)](#), [Point2D::i](#), [Point2D::j](#), and [Dims::w\(\)](#).

Referenced by [computeWhenResetPos\(\)](#), and [saccade\(\)](#).

**10.165.3.8 void ModelEyeHeadController::saccade (const [Point2DT](#) & *target*)**

Initiate a realistic saccade.

Definition at line 2294 of file SaccadeControllers.C.

References [SaccadeController::beginSaccade\(\)](#), [Point2D::i](#), [Point2D::j](#), [pix2deg\(\)](#), and [sqrt\(\)](#).

## 10.165.4 Member Data Documentation

### 10.165.4.1 [ModelParam<Dims>](#) [ModelEyeHeadController::itsDims](#) [protected]

[Image](#) size.

Definition at line 834 of file SaccadeControllers.H.

### 10.165.4.2 [ModelParam<double>](#) [ModelEyeHeadController::itsPPD](#) [protected]

pixels per degree

Definition at line 833 of file SaccadeControllers.H.

The documentation for this class was generated from the following files:

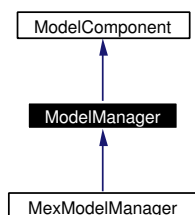
- [SaccadeControllers.H](#)
- [SaccadeControllers.C](#)

## 10.166 ModelManager Class Reference

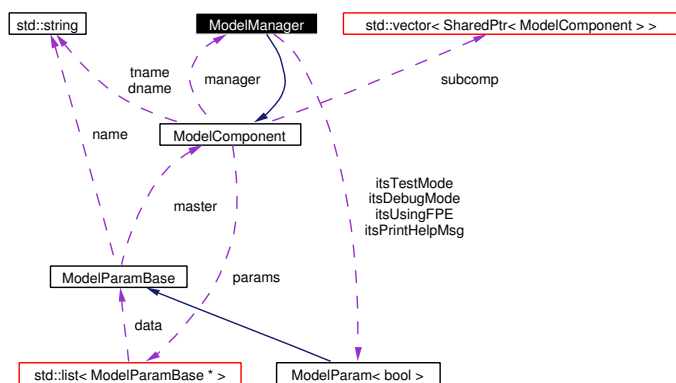
Manages a collection of [ModelComponent](#) objects forming a model.

```
#include <ModelManager.H>
```

Inheritance diagram for ModelManager:



Collaboration diagram for ModelManager:



### Public Member Functions

- `bool setModelParamString (const std::string &name, const std::string &value, const bool recurse=false, const bool warn=true)`  
*Intercept people changing our ModelParam's.*

### Constructors and Destructors

- `ModelManager (const std::string &descrName="The Model", const std::string &tagName="model", const bool loadCfgOpt=true, const bool saveCfgOpt=true, const bool autoLoadCfg=true)`

Default constructor.

- virtual `~ModelManager ()`  
*Virtual destructor ensures proper destruction of derived classes.*
- bool `debugMode ()` const  
*Returns true if we are running in debug mode.*
- bool `usingFPE ()` const  
*Returns true if we are using floating-point exceptions.*
- bool `testMode ()` const  
*Returns true if we are in test mode as per `-test-mode`.*

### ModelComponent management functions

- void `registerComponent (const SharedPtr< ModelComponent > component)`  
*Register a component with the manager.*
- void `unregisterComponent (const SharedPtr< ModelComponent > component)`  
*Unregister a component with the manager.*
- void `destroyComponent (const ModelComponent *c)`  
*ModelComponent objects will call this upon destruction.*

### Command-line options

- void `requestOption (ModelComponent *c, ModelParamBase &p, const bool useMyVal=false)`  
*ModelComponent objects may request a command-line option here.*
- void `unRequestOption (ModelComponent *c, ModelParamBase &p)`  
*ModelComponents may request the removal from the command-line options.*
- void `requestOptionAlias (const std::string name)`  
*Users may request model option aliases.*
- bool `setOptionValString (const std::string &name, const std::string &val)`  
*Set an option value.*
- std::string `getOptionValString (const std::string &name)`  
*Get an option value.*

- bool `parseCommandLine` (const int argc, const char \*\*argv, const char \*usage, const int minarg, const int maxarg)  
*Parse all command-line options and set our [ModelComponent](#) params.*
- uint `numExtraArgs` () const  
*Get the number of available extra args.*
- std::string `getExtraArg` (const uint num) const  
*Get an extra arg, by number.*
- void `printUsage` (const char \*procname, const char \*usage)  
*Print short usage message.*
- void `printHelp` (const char \*errmsg=NULL, const char \*err2=NULL)  
*Print a categorized help message depending on requested options.*
- void `exportOptions` (const int whatToExport, const bool recurse=true)  
*Overload of the [ModelComponent](#) function.*

#### Logging to a text file (`-textlog=XX` option)

- void `textLog` (const std::string &message)  
*Log a message to our textlog file defined via `-textlog=XX` option.*
- void `textLog` (const char \*message,...)  
*Log a message to our textlog file defined via `-textlog=XX` option.*

#### Persistent `ModelParam<T>` management

- bool `loadConfig` ()  
*Load all our `ModelParam<T>` values from file.*
- bool `loadConfig` (const std::string &fname)  
*Explicitly load all our `ModelParams` from file.*
- bool `saveConfig` () const  
*Save all our `ModelParams` to file.*
- bool `saveConfig` (const std::string &fname) const  
*Explicitly save all our `ModelParams` to file.*

## Protected Member Functions

- int **findOption** (const [std::string](#) &name) const
- [std::string](#) **formatHelpString** (const [std::string](#) &h, const unsigned int spc1, const unsigned int spc2, const unsigned int ncols) const
- void **start1** ()  
*This is called from within [start\(\)](#) before the subcomponents start.*
- void **stop2** ()  
*This is called from within [stop\(\)](#) after the subcomponents have stopped.*
- bool **parseCommandLineCore** (const int argc, const char \*\*argv)
- bool **parseCommandLineCore** (const char \*args)
- virtual void **exitProgram** (int return\_code=0)  
*virtual function for exiting*

## Protected Attributes

- [ModelParam](#)< bool > **itsPrintHelpMsg**
- [ModelParam](#)< bool > **itsDebugMode**
- [ModelParam](#)< bool > **itsUsingFPE**
- [ModelParam](#)< bool > **itsTestMode**
- [ModelParam](#)< [std::string](#) > **itsTextLogFile**
- [ModelParam](#)< [std::string](#) > **LoadConfigFile**
- [ModelParam](#)< [std::string](#) > **SaveConfigFile**
- bool **autoLoadConfig**
- [std::vector](#)< [std::string](#) > **extraargs**
- [ModelOption](#) \* **opt**
- int **numopt**
- bool **didExport**
- bool **didSave**
- [std::ofstream](#) \* **itsTextLogStream**

### 10.166.1 Detailed Description

Manages a collection of [ModelComponent](#) objects forming a model.

ModelManager should be implemented once in each executable, and will facilitate the management of a collection of [ModelComponent](#) objects, load/save of tunable [ModelParam](#)<T> parameters that may be in those objects, and generation/parsing of command-line options. [ModelComponent](#) objects must manually register with the ModelManager to fall under its management. Typically, when a [ModelComponent](#) has

subcomponents, only the top-level component should be registered with the ModelManager, as all ModelParam<T> access functions will recurse through the ModelComponent's subcomponents. See test-model.C and test-audioGrab.C for simple examples of how to write executables that are managed by ModelManager.

Definition at line 56 of file ModelManager.H.

## 10.166.2 Constructor & Destructor Documentation

**10.166.2.1 ModelManager::ModelManager** (const [std::string](#) & *descrName* = "The Model", const [std::string](#) & *tagName* = "model", const bool *loadCfgOpt* = true, const bool *saveCfgOpt* = true, const bool *autoLoadCfg* = true)

Default constructor.

### Parameters:

*descrName* descriptine name  
*tagName* tag name (see [ModelComponent.H](#))  
*loadCfgOpt* create a --load-config-from=X option if true  
*saveCfgOpt* create a --save-config-to=X option if true  
*autoLoadCfg* try to load config from ~/.execname if true

Definition at line 55 of file ModelManager.C.

References [AllModelOptions](#), [ModelOptionDef::defval](#), [fpIsEnabled\(\)](#), [requestOption\(\)](#), and [ModelOptionDef::typ](#).

**10.166.2.2 ModelManager::~~ModelManager** () [virtual]

Virtual destructor ensures proper destruction of derived classes.

Definition at line 104 of file ModelManager.C.

References [ModelComponent::destroyManager\(\)](#).

## 10.166.3 Member Function Documentation

**10.166.3.1 bool ModelManager::debugMode** () const

Returns true if we are running in debug mode.

Debug mode can be invoked via the --debug command-line option. Programs can use this function to do additional debug outputs if the manager is in debug mode. In addition, we will generate a debug printout of all model parameters just before [start\(\)](#) and we will force the log level to LOG\_DEBUG (see [log.H](#)) if debug mode is on.



Definition at line 787 of file ModelManager.C.

References `ModelParam< bool >::getVal()`.

Referenced by `ComplexChannel::combineOutputs()`, `SingleChannel::combineSubMaps()`, `ModelComponent::exportOptions()`, `VisualCortex::getOutput()`, `SingleChannel::getRawCSmap()`, `ModelComponent::ModelComponent()`, `SingleChannel::postProcessMap()`, `BeoChip::run()`, `ModelComponent::start()`, `ModelComponent::stop()`, `BeoChip::writeByte()`, and `ModelComponent::~~ModelComponent()`.

### 10.166.3.2 `void ModelManager::destroyComponent (const ModelComponent * c)`

[ModelComponent](#) objects will call this upon destruction.

This should not be called manually; [ModelComponent](#) objects will call this upon destruction. This will remove the corresponding [ModelComponent](#) from our list of command-line option customers and from our list of registered components (both as applicable).

Definition at line 139 of file ModelManager.C.

Referenced by `ModelComponent::~~ModelComponent()`.

### 10.166.3.3 `void ModelManager::exitProgram (int return_code = 0) [protected, virtual]`

virtual function for exiting

Reimplemented in [MexModelManager](#).

Definition at line 621 of file ModelManager.C.

Referenced by `printHelp()`, and `printUsage()`.

### 10.166.3.4 `void ModelManager::exportOptions (const int whatToExport, const bool recurse = true) [virtual]`

Overload of the [ModelComponent](#) function.

This just behaves like the [ModelComponent exportOptions\(\)](#), plus we keep track of whether this was called or not, so that we may call it automatically from within [parseCommandLine\(\)](#), if [exportOptions\(\)](#) has not been called before.

Reimplemented from [ModelComponent](#).

Definition at line 497 of file ModelManager.C.

References `ModelComponent::exportOptions()`.

Referenced by `parseCommandLine()`, `MexModelManager::parseCommandLine()`, and `V4LgrabberLight::V4LgrabberLight()`.

#### 10.166.3.5 `std::string ModelManager::getExtraArg (const uint num) const`

Get an extra arg, by number.

Definition at line 508 of file `ModelManager.C`.

References `uint`.

Referenced by `main()`.

#### 10.166.3.6 `std::string ModelManager::getOptionValString (const std::string & name)`

Get an option value.

The value returned here is the value currently held by the `ModelManager`. This value is updated in three cases: 1) when a new component requests the option using `requestOption()` with `useMyVal` true; 2) when `setOptionValString()` is called, and 3) when the command-line arguments are parsed by `ParseCommandLine`, if a value for this option has been specified in the command line. If `ModelComponent` objects modify their local `ModelParam` values for their local parameter with this name, the value returned here will not be affected.

Definition at line 236 of file `ModelManager.C`.

#### 10.166.3.7 `bool ModelManager::loadConfig (const std::string & fname)`

Explicitly load all our `ModelParams` from file.

This function will always load the params from the given file, even if `-load-config-from=X` was not specified.

Definition at line 664 of file `ModelManager.C`.

References `ParamMap::load()`, and `ModelComponent::readParamsFrom()`.

#### 10.166.3.8 `bool ModelManager::loadConfig ()`

Load all our `ModelParam<T>` values from file.

This function is automatically called during `parseCommandLine()` if `-load-config-from=XX` option is encountered or `autoload` was specified in our constructor (thus ensuring that command-line options encountered after the option specifying which config file to load are applied after the config file has been loaded). It may also be

called manually. This function is a no-op if no filename is in our loadConfigFile internal [ModelParam](#) (this parameter can be set by the `--load-config-from=X` command-line option). Otherwise, load the file (which should be a [ParamMap](#)) and configure all our parameters. Returns true on success.

Definition at line 652 of file ModelManager.C.

References [ParamMap::load\(\)](#), and [ModelComponent::readParamsFrom\(\)](#).

### 10.166.3.9 [uint](#) ModelManager::numExtraArgs () const

Get the number of available extra args.

Definition at line 504 of file ModelManager.C.

References [uint](#).

Referenced by [main\(\)](#).

### 10.166.3.10 [bool](#) ModelManager::parseCommandLine (const int argc, const char \*\* argv, const char \* usage, const int minarg, const int maxarg)

Parse all command-line options and set our [ModelComponent](#) params.

This function will create a set of command-line options from our current collection of ModelComponents. It will then parse the command-line arguments and set the various option values accordingly, using a call to [setOptionValString\(\)](#). Any left-over args (i.e., not formatted as options) will be available through [getExtraArg\(\)](#), and if the number of such extra args is not within the specified minarg/maxarg bounds, a usage message and optionlist will be printed out (also printed out if a `'-help'` option is encountered). To decide which options should be exported by your various [ModelComponent](#) objects, typically you will call [exportOptions\(\)](#) on the manager, which will propagate the call to all registered components. Here we have an additional default behavior that if [exportOptions\(\)](#) has not been called on us, we will call it with an order to export all options, just before we start parsing the command-line. So, if you want to export all options, you don't have to do anything; if you want to only export special classes of options, then you will need to call [exportOptions\(\)](#) manually some time before you parse the command-line. Returns true upon fully successful parse.

#### Parameters:

**argc** the number of command-line args

**argv** the command-line args

**usage** a textual description of the expected usage format, describing only what the non-option arguments should be (e.g., "`<input.ppm> <output.ppm>`")

**minarg** the minimum number of non-optional arguments

**maxarg** the maximum number of non-optional args, or -1 if unbounded

Definition at line 246 of file ModelManager.C.

References `exportOptions()`, `ParamMap::load()`, `OPTEXP_ALL`, `printUsage()`, `ModelComponent::readParamsFrom()`, and `uint`.

Referenced by `main()`.

#### 10.166.3.11 **void ModelManager::printHelp** (const char \* *errmsg* = NULL, const char \* *err2* = NULL)

Print a categorized help message depending on requested options.

This is automatically called during `parseCommandLine()` if necessary

Definition at line 527 of file ModelManager.C.

References `AllModelOptionCats`, `AllModelOptions`, `convertFromString()`, `ModelOptionDef::defval`, `ModelOptionDef::descr`, `exitProgram()`, `ModelOptionDef::longoptname`, `ModelOptionDef::name`, `NBMODELOPTIONCATEGS`, `ModelOptionDef::shortoptname`, `ModelOptionDef::typ`, `uint`, and `ModelOptionDef::validvals`.

#### 10.166.3.12 **void ModelManager::printUsage** (const char \* *procname*, const char \* *usage*)

Print short usage message.

This is automatically called during `parseCommandLine()` if necessary

Definition at line 518 of file ModelManager.C.

References `exitProgram()`.

Referenced by `parseCommandLine()`.

#### 10.166.3.13 **void ModelManager::registerComponent** (const [SharedPtr](#) < [ModelComponent](#) > *component*)

Register a component with the manager.

This is only a different name for `ModelComponent`'s `addSubComponent()`, just to make `ModelManager` feel special.

Definition at line 121 of file ModelManager.C.

References `ModelComponent::addSubComponent()`.

Referenced by `MbariResultViewer::MbariResultViewer()`, and `V4LgrabberLight::V4LgrabberLight()`.

### 10.166.3.14 void ModelManager::requestOption ([ModelComponent](#) \* c, [ModelParamBase](#) & p, const bool useMyVal = false)

[ModelComponent](#) objects may request a command-line option here.

#### Parameters:

- c* the [ModelComponent](#) object requesting the option
- p* the [ModelParam](#) data member of *c* corresponding to the option
- useMyVal* if true, we will do a [setOptionValString\(\)](#) using the current value of the parameter, so that this value will be propagated to all [ModelComponent](#) objects that have requested this option and will become the new command-line default value. If false, the [ModelParam](#) *p* will be updated to the current value of the option. So, use *useMyVal*=true if your component wishes to push its current parameter value as the default for everyone using this option, and use false if your component wishes to use the current default instead.

Definition at line 157 of file ModelManager.C.

References [ModelParamBase::getName\(\)](#), [ModelParamBase::getValString\(\)](#), [ModelComponent::setModelParamString\(\)](#), and [setOptionValString\(\)](#).

Referenced by [ModelComponent::doRequestOption\(\)](#), [WinnerTakeAllConfigurator::exportOptions\(\)](#), [WinnerTakeAll::exportOptions\(\)](#), [VisualCortexEyeMvt::exportOptions\(\)](#), [VisualCortexConfigurator::exportOptions\(\)](#), [VisualCortex::exportOptions\(\)](#), [VisualBuffer::exportOptions\(\)](#), [VCC4::exportOptions\(\)](#), [TCPcommunicator::exportOptions\(\)](#), [TaskRelevanceMapKillN::exportOptions\(\)](#), [TaskRelevanceMapKillStatic::exportOptions\(\)](#), [TaskRelevanceMapStd::exportOptions\(\)](#), [TaskRelevanceMapConfigurator::exportOptions\(\)](#), [TaskRelevanceMap::exportOptions\(\)](#), [SoxChannel::exportOptions\(\)](#), [SockServ::exportOptions\(\)](#), [SingleChannelSurprise::exportOptions\(\)](#), [SingleChannelBeoServer::exportOptions\(\)](#), [SingleChannel::exportOptions\(\)](#), [SimulationViewerStd::exportOptions\(\)](#), [SimulationViewerEyeMvt2::exportOptions\(\)](#), [SimulationViewerEyeMvt::exportOptions\(\)](#), [SimulationViewerCompress::exportOptions\(\)](#), [SimulationViewerConfigurator::exportOptions\(\)](#), [ShapeEstimator::exportOptions\(\)](#), [SaliencyMapFast::exportOptions\(\)](#), [SaliencyMapStd::exportOptions\(\)](#), [SaliencyMapConfigurator::exportOptions\(\)](#), [SaliencyMap::exportOptions\(\)](#), [ModelEyeHeadController::exportOptions\(\)](#), [ModelWrapperSaccadeController::exportOptions\(\)](#), [VisBufSaccadeController::exportOptions\(\)](#), [EyeTrackerSaccadeController::exportOptions\(\)](#), [MonkeySaccadeController::exportOptions\(\)](#), [Monkey2SaccadeController::exportOptions\(\)](#), [MonkeyEyeHeadController::exportOptions\(\)](#), [ThresholdFrictionSaccadeController::exportOptions\(\)](#), [ThresholdSaccadeController::exportOptions\(\)](#), [FrictionSaccadeController::exportOptions\(\)](#), [FixedSaccadeController::exportOptions\(\)](#), [SaccadeControllerConfigurator::exportOptions\(\)](#), [RetinaStd::exportOptions\(\)](#), [RetinaConfigurator::exportOptions\(\)](#), [PsychoDisplay::exportOptions\(\)](#), [MPEGStream::exportOptions\(\)](#), [MbariResultViewer::exportOptions\(\)](#), [InferoTemporalConfigurator::exportOptions\(\)](#), [InferoTemporal::exportOptions\(\)](#), [Image](#)

Display::exportOptions(), GistEstimatorConfigurator::exportOptions(), FrameSeries::exportOptions(), FrameGrabberConfigurator::exportOptions(), FrameGrabber::exportOptions(), DirectFeedChannel::exportOptions(), ComplexChannel::exportOptions(), DummyChannel::exportOptions(), ScorrChannel::exportOptions(), TcorrChannel::exportOptions(), StereoChannel::exportOptions(), EndPointChannel::exportOptions(), XJunctionChannel::exportOptions(), TJunctionChannel::exportOptions(), LJunctionChannel::exportOptions(), JunctionChannel::exportOptions(), MotionChannel::exportOptions(), DirectionChannel::exportOptions(), MultiColorBandChannel::exportOptions(), OrientationChannel::exportOptions(), ColorChannel::exportOptions(), GaborChannel::exportOptions(), ChannelBase::exportOptions(), Brain::exportOptions(), Beowulf::exportOptions(), AudioMixer::exportOptions(), AudioGrabber::exportOptions(), AttentionGuidanceMapConfigurator::exportOptions(), AttentionGuidanceMap::exportOptions(), ModelManager(), and Beowulf::setModelParamString().

#### 10.166.3.15 void ModelManager::requestOptionAlias (const std::string name)

Users may request model option aliases.

##### Parameters:

**name** the name of the option alias, from [ModelOptionDefs.C](#)

Definition at line 195 of file ModelManager.C.

#### 10.166.3.16 bool ModelManager::saveConfig (const std::string & fname) const

Explicitly save all our ModelParams to file.

This function will always save the params to the given file, even if `--save-config-to=X` was not specified.

Definition at line 683 of file ModelManager.C.

References [ParamMap::format\(\)](#), and [ModelComponent::writeParamsTo\(\)](#).

#### 10.166.3.17 bool ModelManager::saveConfig () const

Save all our ModelParams to file.

This function is a no-op if no filename is in our `saveConfigFile` internal [ModelParam](#) (this parameter can be set by the `--save-config-to=X` command-line option). Otherwise, save all our params to the file as a [ParamMap](#). Returns true on success. You should typically call this either some time before you [start\(\)](#) your model, or after you have [stop\(\)](#)'ed it. NOTE: There is a default behavior which is to call this as the first thing in [start\(\)](#), i.e., just before all our registered components will be started, unless you have manually called it yourself earlier, in which case it is not called again.

Definition at line 673 of file ModelManager.C.

Referenced by start1().

**10.166.3.18** `bool ModelManager::setModelParamString (const std::string & name, const std::string & value, const bool recurse = false, const bool warn = true) [virtual]`

Intercept people changing our ModelParam's.

See [ModelComponent.H](#); as parsing the command-line or reading a config file sets our name, we'll also here instantiate a controller of the proper type (and export its options)

Reimplemented from [ModelComponent](#).

Definition at line 734 of file ModelManager.C.

References [fpExceptionsOff\(\)](#), [fpExceptionsOn\(\)](#), [ModelParamBase::getName\(\)](#), [ModelParam< bool >::getVal\(\)](#), [MYLOGVERB](#), [ModelComponent::setModelParamString\(\)](#), and [setOptionValString\(\)](#).

**10.166.3.19** `bool ModelManager::setOptionValString (const std::string & name, const std::string & val)`

Set an option value.

The [ModelParam](#) objects in all [ModelComponent](#) objects that have requested this option will be updated with the new value. Will throw a fatal error if the model has been started (see [ModelComponent::start\(\)](#)).

Definition at line 213 of file ModelManager.C.

References [AllModelOptions](#).

Referenced by [main\(\)](#), [ModelWrapperSaccadeController::ModelWrapperSaccadeController\(\)](#), [Monkey2SaccadeController::Monkey2SaccadeController\(\)](#), [requestOption\(\)](#), [setModelParamString\(\)](#), [SimulationViewerCompress::SimulationViewerCompress\(\)](#), [SimulationViewerEyeMvt::SimulationViewerEyeMvt\(\)](#), [SimulationViewerEyeMvt2::SimulationViewerEyeMvt2\(\)](#), and [V4LgrabberLight::V4LgrabberLight\(\)](#).

**10.166.3.20** `void ModelManager::start1 () [protected, virtual]`

This is called from within [start\(\)](#) before the subcomponents start.

Reimplemented from [ModelComponent](#).

Definition at line 703 of file ModelManager.C.

References `ModelParam< bool >::getVal()`, `ModelComponent::printout()`, and `save-Config()`.

#### 10.166.3.21 `void ModelManager::stop2 ()` [protected, virtual]

This is called from within `stop()` after the subcomponents have stopped.

Reimplemented from `ModelComponent`.

Definition at line 775 of file `ModelManager.C`.

#### 10.166.3.22 `bool ModelManager::testMode ()` const

Returns true if we are in test mode as per `-test-mode`.

Various components will adjust their behavior in various ways when in test mode. The goal is to provide an exactly reproducible, batch-oriented execution of each program, to be used in the test suite. At the manager level, we turn off `-use-random` and initialize random numbers to a fixed seed (in case some components still use random numbers while `-use-random` is off), while we initialize the random numbers to a time and PID-dependent seed otherwise. Do a `grep` for the name of this function on other components to see if they use it (e.g., `FrameSeries` will turn off all displays when in test mode).

Definition at line 795 of file `ModelManager.C`.

References `ModelParam< bool >::getVal()`.

Referenced by `FrameSeries::okToDisplay()`.

#### 10.166.3.23 `void ModelManager::textLog (const char * message, ...)`

Log a message to our textlog file defined via `-textlog=XX` option.

This version is for `printf()`-like message syntax. If the option was not specified in the command line, this is a no-op.

Definition at line 809 of file `ModelManager.C`.

#### 10.166.3.24 `void ModelManager::textLog (const std::string & message)`

Log a message to our textlog file defined via `-textlog=XX` option.

This version is for C++ formatted message. If the option was not specified in the command line, this is a no-op.

Definition at line 799 of file `ModelManager.C`.

Referenced by `Brain::evolve()`, `InferoTemporal::input()`, `Brain::processInput()`, `InputFrameSeries::readGray()`, `InputFrameSeries::readRGB()`, `InputFrameSeries::start2()`,



FrameSeries::update(), OutputFrameSeries::writeFloat(), and OutputFrameSeries::writeGray().

#### 10.166.3.25 void ModelManager::unregisterComponent (const [SharedPtr< ModelComponent >](#) component)

Unregister a component with the manager.

This is only a different name for ModelComponent's [removeSubComponent\(\)](#), just to make ModelManager feel special. If you have a component that has requested options from the manager and is about to be destroyed, be sure to call this first so that the manager will not try to set the option values on your component.

Definition at line 125 of file ModelManager.C.

References [SharedPtr< T >::get\(\)](#), and [ModelComponent::removeSubComponent\(\)](#).

#### 10.166.3.26 void ModelManager::unRequestOption ([ModelComponent \\* c](#), [ModelParamBase & p](#))

ModelComponents may request the removal from the command-line options.

##### Parameters:

- c* the [ModelComponent](#) object requesting the option
- p* the [ModelParam](#) data member of *c* corresponding to the option

Definition at line 181 of file ModelManager.C.

References [ModelParamBase::getName\(\)](#).

Referenced by [MexModelManager::MexModelManager\(\)](#).

#### 10.166.3.27 bool ModelManager::usingFPE () const

Returns true if we are using floating-point exceptions.

Floating point exceptions are on by default, but may be turned off via the command-line option `-nouse-fpe`.

Reimplemented in [MexModelManager](#).

Definition at line 791 of file ModelManager.C.

References [ModelParam< bool >::getVal\(\)](#).

The documentation for this class was generated from the following files:

- [ModelManager.H](#)
- [ModelManager.C](#)

## 10.167 ModelOptionDef Struct Reference

Specification of a known model option.

```
#include <ModelOptionDefs.H>
```

### Public Attributes

- [ModelOptionType](#) `typ`  
*Type of the option.*
- `const char *` [name](#)  
*Option name.*
- `const` [ModelOptionCateg](#) `categ`  
*Option category.*
- `const char *` [descr](#)  
*Description of what option does.*
- `const char *` [longoptname](#)  
*Long option name (without leading "-").*
- `char` [shortoptname](#)  
*Short option name (without leading "-").*
- `const char *` [validvals](#)  
*Textual description of valid values, or NULL.*
- `const char *` [defval](#)  
*Default value in text representation.*

### 10.167.1 Detailed Description

Specification of a known model option.

This struct is used to build a list of all known model options in ModelOptions.C

Definition at line 100 of file ModelOptionDefs.H.

## 10.167.2 Member Data Documentation

### 10.167.2.1 `const ModelOptionCateg ModelOptionDef::categ`

Option category.

Definition at line 104 of file ModelOptionDefs.H.

### 10.167.2.2 `const char* ModelOptionDef::defval`

Default value in text representation.

Definition at line 109 of file ModelOptionDefs.H.

Referenced by ModelManager::ModelManager(), and ModelManager::printHelp().

### 10.167.2.3 `const char* ModelOptionDef::descr`

Description of what option does.

Definition at line 105 of file ModelOptionDefs.H.

Referenced by ModelManager::printHelp().

### 10.167.2.4 `const char* ModelOptionDef::longoptname`

Long option name (without leading "-").

Definition at line 106 of file ModelOptionDefs.H.

Referenced by ModelManager::printHelp().

### 10.167.2.5 `const char* ModelOptionDef::name`

Option name.

Definition at line 103 of file ModelOptionDefs.H.

Referenced by ModelManager::printHelp().

### 10.167.2.6 `char ModelOptionDef::shortoptname`

Short option name (without leading "-").

Definition at line 107 of file ModelOptionDefs.H.

Referenced by ModelManager::printHelp().

**10.167.2.7   [ModelOptionType](#) [ModelOptionDef::typ](#)**

Type of the option.

Definition at line 102 of file ModelOptionDefs.H.

Referenced by ModelManager::ModelManager(), and ModelManager::printHelp().

**10.167.2.8   `const char*` [ModelOptionDef::validvals](#)**

Textual description of valid values, or NULL.

Definition at line 108 of file ModelOptionDefs.H.

Referenced by ModelManager::printHelp().

The documentation for this struct was generated from the following file:

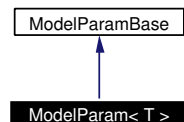
- [ModelOptionDefs.H](#)

## 10.168 ModelParam< T > Class Template Reference

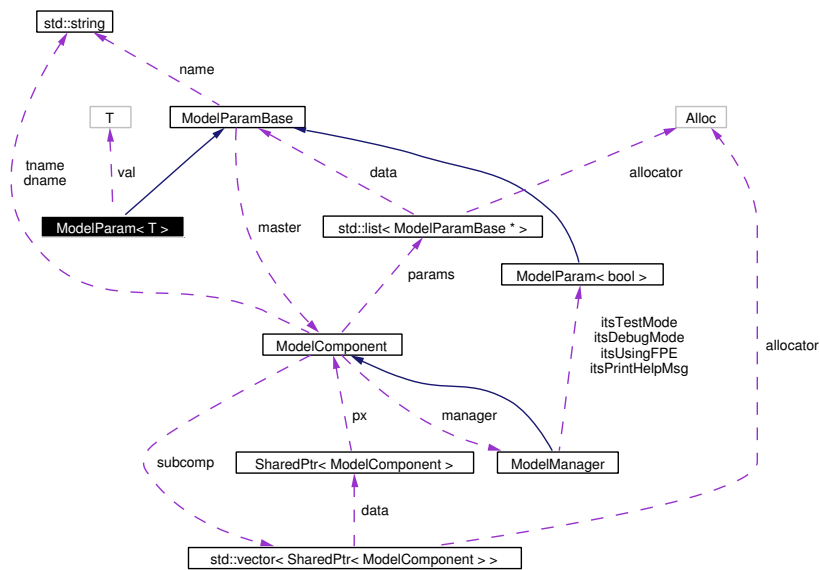
ModelParam<T> definitions.

```
#include <ModelParam.H>
```

Inheritance diagram for ModelParam< T >:



Collaboration diagram for ModelParam< T >:



## Public Member Functions

### Constructors and Destructors

- [ModelParam](#) ()  
*Uninitialized constructor.*
- [ModelParam](#) ([ModelComponent](#) \*mast, const [std::string](#) &nam, const T &initval)

*Full constructor.*

- void `init` (`ModelComponent` \*mast, const `std::string` &nam, const T &init-val)

*Initialization.*

### Access functions

- void `setVal` (const T &v)  
*Set the value.*
- T `getVal` () const  
*Get the value.*
- bool `setValString` (const `std::string` &textval)  
*Set the value from a string representation.*
- `std::string` `getValString` () const  
*Get the value as a string.*

## 10.168.1 Detailed Description

`template<class T> class ModelParam< T >`

`ModelParam<T>` definitions.

`ModelParam<T>` is a template class that derives from `ModelParamBase` and implements a tunable parameter of type T. To the `ModelParamBase` interface, `ModelParam<T>` adds `getVal()` and `setVal()` for direct access to the parameter value.

Definition at line 154 of file `ModelParam.H`.

## 10.168.2 Constructor & Destructor Documentation

**10.168.2.1** `template<class T> ModelParam< T >::ModelParam ()`

Uninitialized constructor.

You will need to call `init()` before any other method

Definition at line 148 of file `ModelParam.C`.

**10.168.2.2** `template<class T> ModelParam< T >::ModelParam  
(ModelComponent * mast, const std::string & nam, const T &  
initval)`

Full constructor.

Same as [ModelParamBase](#) plus an initial parameter value

Definition at line 153 of file [ModelParam.C](#).

### 10.168.3 Member Function Documentation

**10.168.3.1** `template<class T> T ModelParam< T >::getVal () const`

Get the value.

Definition at line 171 of file [ModelParam.C](#).

**10.168.3.2** `template<class T> std::string ModelParam< T >::getValString ()  
const [virtual]`

Get the value as a string.

Implements [ModelParamBase](#).

Definition at line 176 of file [ModelParam.C](#).

References [convertToString\(\)](#).

**10.168.3.3** `template<class T> void ModelParam< T >::init (ModelComponent  
* mast, const std::string & nam, const T & initval)`

Initialization.

Same as [ModelParamBase](#) plus an initial parameter value

Definition at line 160 of file [ModelParam.C](#).

References [ModelParamBase::init\(\)](#).

**10.168.3.4** `template<class T> void ModelParam< T >::setVal (const T & v)`

Set the value.

Definition at line 166 of file [ModelParam.C](#).

**10.168.3.5** `template<class T> bool ModelParam< T >::setValString (const std::string & textval) [virtual]`

Set the value from a string representation.

Implements [ModelParamBase](#).

Definition at line 181 of file `ModelParam.C`.

References `convertFromString()`.

The documentation for this class was generated from the following files:

- [ModelParam.H](#)
- [ModelParam.C](#)

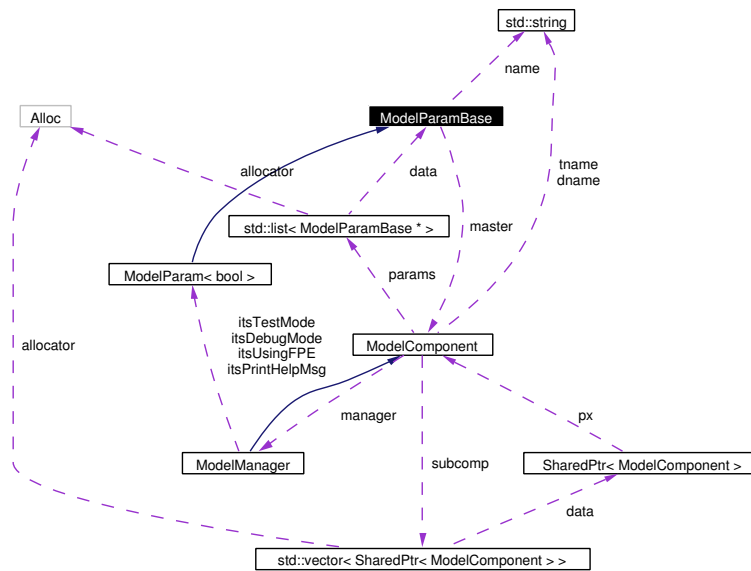


## 10.169 ModelParamBase Class Reference

ModelParamBase is a persistent [ModelComponent](#) parameter base class.

```
#include <ModelParam.H>
```

Inheritance diagram for ModelParamBase: Collaboration diagram for ModelParamBase:



## Public Member Functions

### Constructors and Destructors

- [ModelParamBase](#) ()  
*Uninitialized constructor.*
- [ModelParamBase](#) ([ModelComponent](#) \*mast, const [std::string](#) &nam)  
*Initialized constructor.*
- virtual void [init](#) ([ModelComponent](#) \*mast, const [std::string](#) &nam)  
*Initialization of an uninitialized ModelParamBase.*
- bool [initialized](#) () const  
*Return true if we have been initialized.*
- virtual [~ModelParamBase](#) ()

*Destructor.*

### Basic access functions

- `std::string getName () const`  
*get the ModelParamBase's name*
- virtual `bool setValString (const std::string &textval)=0`  
*Set the parameter value from a textual representation.*
- virtual `std::string getValString () const=0`  
*Get current value as a string (must be implemented by derived classes).*

### Input/Output functions

- virtual `void printout (std::ostream &s, const std::string &prefix="") const`  
*Print out our name and contents, mostly for debugging.*
- `void writeTo (ParamMap &pmap) const`  
*Write parameter value to ParamMap.*
- `void readFrom (const ParamMap &pmap, const bool noerr=true)`  
*Get parameter value from ParamMap.*

### Protected Attributes

- `ModelComponent * master`  
*keep track of our master*
- `std::string name`  
*the parameter name as used in ParamMap*

## 10.169.1 Detailed Description

ModelParamBase is a persistent [ModelComponent](#) parameter base class.

Various model parameter types can be derived from this base, see the `ModelParam<T>` template class definition. ModelParamBase is persistent in the sense that it can be saved/loaded into a [ParamMap](#), which itself provides disk save/load capabilities. ModelParamBase is intended to be used as a data member in a [ModelComponent](#) (or

derived) object. [ModelComponent](#) provides an interface by which ModelParamBase derivatives may also be tuned via command-line options.

Definition at line 56 of file ModelParam.H.

## 10.169.2 Constructor & Destructor Documentation

### 10.169.2.1 ModelParamBase::ModelParamBase ()

Uninitialized constructor.

The ModelParamBase created using this constructor will be unusable until it has been configured using [init\(\)](#). This constructor is provided for the creation of ModelParamBase arrays.

Definition at line 71 of file ModelParam.C.

### 10.169.2.2 ModelParamBase::ModelParamBase ([ModelComponent](#) \* *mast*, const [std::string](#) & *nam*)

Initialized constructor.

#### Parameters:

*mast* the [ModelComponent](#) which we are attached to

*nam* the name of the ModelParamBase, that will be used to store its value in a [ParamMap](#) and fetch command-line option configuration data.

Definition at line 76 of file ModelParam.C.

References [ModelComponent::registerParam\(\)](#).

### 10.169.2.3 ModelParamBase::~~ModelParamBase () [virtual]

Destructor.

Definition at line 98 of file ModelParam.C.

References [ModelComponent::unregisterParam\(\)](#).

## 10.169.3 Member Function Documentation

### 10.169.3.1 [std::string](#) ModelParamBase::getName () const

get the ModelParamBase's name

Definition at line 105 of file ModelParam.C.

Referenced by `ModelManager::requestOption()`, `SoxChannel::setModelParamString()`, `SimulationViewerCompress::setModelParamString()`, `ModelManager::setModelParamString()`, `MbariResultViewer::setModelParamString()`, `StereoChannel::setModelParamString()`, `EndPointChannel::setModelParamString()`, `XJunctionChannel::setModelParamString()`, `TJunctionChannel::setModelParamString()`, `LJunctionChannel::setModelParamString()`, `MotionChannel::setModelParamString()`, `MultiColorBandChannel::setModelParamString()`, `OrientationChannel::setModelParamString()`, `Beowulf::setModelParamString()`, and `ModelManager::unRequestOption()`.

### 10.169.3.2 `virtual std::string ModelParamBase::getValString () const` [pure virtual]

Get current value as a string (must be implemented by derived classes).

Typically, a derived class of type `T` will also implement a method "`T getVal() const`" for more efficient direct access to the parameter value. See the class definition for `ModelParam<T>`

Implemented in `ModelParam< T >`, `ModelParam< ShapeEstimatorMode >`, `ModelParam< PixRGB< byte > >`, `ModelParam< ShapeEstimatorSmoothMethod >`, `ModelParam< double >`, `ModelParam< ColorComputeType >`, `ModelParam< std::string >`, `ModelParam< VisualFeature >`, `ModelParam< BitObjectDrawMode >`, `ModelParam< in_addr >`, `ModelParam< float >`, `ModelParam< FrameRange >`, `ModelParam< int >`, `ModelParam< short int >`, `ModelParam< PyramidType >`, `ModelParam< FrameGrabberMode >`, `ModelParam< LevelSpec >`, `ModelParam< Point2D >`, `ModelParam< IORtype >`, `ModelParam< RasterFileFormat >`, `ModelParam< ImageDisplayType >`, `ModelParam< MaxNormType >`, `ModelParam< bool >`, `ModelParam< Dims >`, `ModelParam< uint >`, and `ModelParam< OrientComputeType >`.

Referenced by `printout()`, `readFrom()`, `ModelManager::requestOption()`, and `writeTo()`.

### 10.169.3.3 `void ModelParamBase::init (ModelComponent * mast, const std::string & nam)` [virtual]

Initialization of an uninitialized `ModelParamBase`.

This will produce a fatal error if the `ModelParamBase` has already been initialized. Should be called before other uses on all `ModelParamBase` objects created with the default constructor.

#### Parameters:

***mast*** the `ModelComponent` that we are attached to

***nam*** the name of the `ModelParamBase` that will be used to store its value in a `ParamMap` and fetch command-line option configuration data.

Definition at line 84 of file ModelParam.C.

References `ModelComponent::registerParam()`.

Referenced by `ModelParam< T >::init()`.

#### 10.169.3.4 `bool ModelParamBase::initialized () const`

Return true if we have been initialized.

Definition at line 94 of file ModelParam.C.

#### 10.169.3.5 `void ModelParamBase::printout (std::ostream & s, const std::string & prefix = "") const [virtual]`

Print out our name and contents, mostly for debugging.

Definition at line 109 of file ModelParam.C.

References `getValString()`.

#### 10.169.3.6 `void ModelParamBase::readFrom (const ParamMap & pmap, const bool noerr = true)`

Get parameter value from [ParamMap](#).

##### Parameters:

*noerr* will not generate an error message if the parameter does not exist in [ParamMap](#).

Definition at line 125 of file ModelParam.C.

References `getValString()`, `ParamMap::queryStringParam()`, and `ModelComponent::setModelParamString()`.

#### 10.169.3.7 `virtual bool ModelParamBase::setValString (const std::string & textual) [pure virtual]`

Set the parameter value from a textual representation.

Returns true on successful parsing. Must be implemented by derived classes. Typically, a derived class of type T will also implement a method "void setVal(const T& v)" for more efficient direct assignment of the parameter value. See the class definition for `ModelParam<T>`

Implemented in [ModelParam< T >](#), [ModelParam< ShapeEstimatorMode >](#), [ModelParam< PixRGB< byte > >](#), [ModelParam< ShapeEstimatorSmoothMethod >](#),

[ModelParam< double >](#), [ModelParam< ColorComputeType >](#), [ModelParam< std::string >](#), [ModelParam< VisualFeature >](#), [ModelParam< BitObjectDrawMode >](#), [ModelParam< in\\_addr >](#), [ModelParam< float >](#), [ModelParam< FrameRange >](#), [ModelParam< int >](#), [ModelParam< short int >](#), [ModelParam< PyramidType >](#), [ModelParam< FrameGrabberMode >](#), [ModelParam< LevelSpec >](#), [ModelParam< Point2D >](#), [ModelParam< IORtype >](#), [ModelParam< RasterFileFormat >](#), [ModelParam< ImageDisplayType >](#), [ModelParam< MaxNormType >](#), [ModelParam< bool >](#), [ModelParam< Dims >](#), [ModelParam< uint >](#), and [ModelParam< OrientComputeType >](#).

#### 10.169.3.8 void ModelParamBase::writeTo ([ParamMap](#) & *pmap*) const

Write parameter value to [ParamMap](#).

Definition at line 118 of file [ModelParam.C](#).

References [getValString\(\)](#), and [ParamMap::putStringParam\(\)](#).

### 10.169.4 Member Data Documentation

#### 10.169.4.1 [ModelComponent\\*](#) [ModelParamBase::master](#) [protected]

keep track of our master

Definition at line 140 of file [ModelParam.H](#).

#### 10.169.4.2 [std::string](#) [ModelParamBase::name](#) [protected]

the parameter name as used in [ParamMap](#)

Definition at line 141 of file [ModelParam.H](#).

The documentation for this class was generated from the following files:

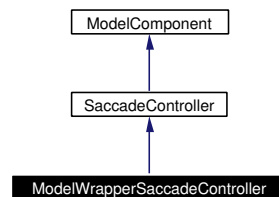
- [ModelParam.H](#)
- [ModelParam.C](#)

## 10.170 ModelWrapperSaccadeController Class Reference

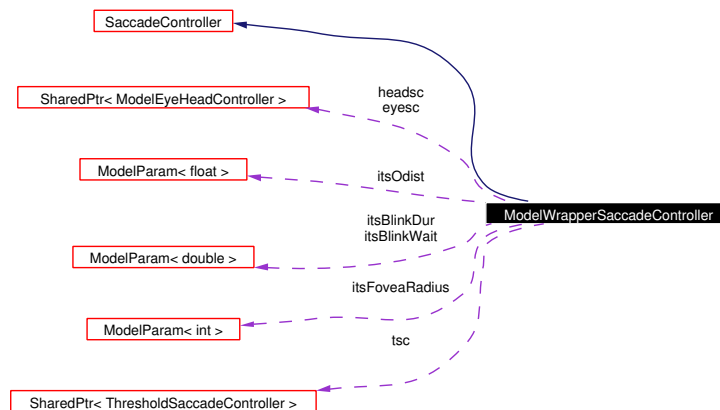
Realistic rhesus monkey saccade controller.

```
#include <SaccadeControllers.H>
```

Inheritance diagram for ModelWrapperSaccadeController:



Collaboration diagram for ModelWrapperSaccadeController:



### Public Member Functions

- [ModelWrapperSaccadeController \(ModelManager &mgr\)](#)  
*Constructor.*
- virtual [~ModelWrapperSaccadeController \(\)](#)  
*Destructor.*
- virtual void [exportOptions](#) (const int whatToExport, const bool recurse=true)

Export some of our [ModelParam](#) parameters as command-line-options.

- virtual void [evolve](#) (const double t)  
*evolve one time step (called, e.g., by [Brain::evolve\(\)](#))*

## Protected Member Functions

- virtual void [computeWhenNewPercept](#) ()  
*This method is called each time a new percept has arrived.*
- virtual void [computeWhenResetPos](#) ()  
*This method is called each time a reset is made.*
- virtual [Point2D](#) [computeWhenNewDecision](#) (const double t)  
*This method is called each time a new decision is requested.*
- virtual [Point2D](#) [computeWhenNewDecisionHead](#) (const double t)  
*This method is called each time a new head decision is requested.*

## Protected Attributes

- [ModelParam](#)< double > [itsBlinkWait](#)  
*wait time before a blink*
- [ModelParam](#)< double > [itsBlinkDur](#)  
*blink duration*
- [ModelParam](#)< float > [itsOdist](#)
- [ModelParam](#)< int > [itsFoveaRadius](#)

### 10.170.1 Detailed Description

Realistic rhesus monkey saccade controller.

This is a hack based on Nitin's [MonkeySaccadeController](#) but decoupling eye and head (so that the head minds its own business without influencing the eye). Not as realistic in the details, but the end result is the same

Definition at line 853 of file SaccadeControllers.H.



## 10.170.2 Constructor & Destructor Documentation

### 10.170.2.1 ModelWrapperSaccadeController::ModelWrapperSaccadeController ([ModelManager](#) & *mgr*)

Constructor.

Definition at line 1915 of file SaccadeControllers.C.

References [ModelComponent::addSubComponent\(\)](#), and [ModelManager::setOptionValString\(\)](#).

### 10.170.2.2 ModelWrapperSaccadeController::~~ModelWrapperSaccadeController () [virtual]

Destructor.

Definition at line 1940 of file SaccadeControllers.C.

## 10.170.3 Member Function Documentation

### 10.170.3.1 [Point2D](#) ModelWrapperSaccadeController::computeWhenNewDecision (const double *t*) [protected, virtual]

This method is called each time a new decision is requested.

Implements [SaccadeController](#).

Definition at line 2142 of file SaccadeControllers.C.

References [Point2D::i](#), and [Point2D::j](#).

### 10.170.3.2 [Point2D](#) ModelWrapperSaccadeController::computeWhenNewDecisionHead (const double *t*) [protected, virtual]

This method is called each time a new head decision is requested.

Reimplemented from [SaccadeController](#).

Definition at line 2162 of file SaccadeControllers.C.

References [Point2D::i](#), and [Point2D::j](#).

**10.170.3.3 void ModelWrapperSaccadeController::computeWhenNewPercept()  
[protected, virtual]**

This method is called each time a new percept has arrived.

Implements [SaccadeController](#).

Definition at line 2045 of file SaccadeControllers.C.

References [SaccadeController::beginSaccade\(\)](#), [SaccadeController::beginSaccadeHead\(\)](#), [SaccadeController::getPreviousPercept\(\)](#), [ModelParam< float >::getVal\(\)](#), [ModelParam< int >::getVal\(\)](#), [Point2D::i](#), [SaccadeController::isInBlink\(\)](#), [Point2D::j](#), [WTAwinner::p](#), and [WTAwinner::t](#).

**10.170.3.4 void ModelWrapperSaccadeController::computeWhenResetPos()  
[protected, virtual]**

This method is called each time a reset is made.

Implements [SaccadeController](#).

Definition at line 2136 of file SaccadeControllers.C.

**10.170.3.5 void ModelWrapperSaccadeController::evolve (const double t)  
[virtual]**

evolve one time step (called, e.g., by [Brain::evolve\(\)](#))

Implements [SaccadeController](#).

Definition at line 1960 of file SaccadeControllers.C.

References [SaccadeController::beginBlink\(\)](#), [SaccadeController::endBlink\(\)](#), [SaccadeController::endSaccade\(\)](#), [SaccadeController::endSaccadeHead\(\)](#), and [SaccadeController::isInBlink\(\)](#).

**10.170.3.6 void ModelWrapperSaccadeController::exportOptions (const int  
whatToExport, const bool recurse = true) [virtual]**

Export some of our [ModelParam](#) parameters as command-line-options.

See the base function in [ModelComponent.H](#) for info.

Reimplemented from [ModelComponent](#).

Definition at line 1944 of file SaccadeControllers.C.

References [ModelComponent::exportOptions\(\)](#), [OPTEXP\\_CORE](#), and [ModelManager::requestOption\(\)](#).

## 10.170.4 Member Data Documentation

### 10.170.4.1 [ModelParam<double>](#) [ModelWrapperSaccadeController::itsBlink-Dur](#) [protected]

blink duration

Definition at line 884 of file SaccadeControllers.H.

### 10.170.4.2 [ModelParam<double>](#) [ModelWrapperSaccadeController::itsBlink-Wait](#) [protected]

wait time before a blink

Definition at line 883 of file SaccadeControllers.H.

The documentation for this class was generated from the following files:

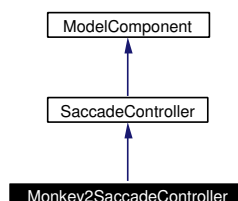
- [SaccadeControllers.H](#)
- [SaccadeControllers.C](#)

## 10.171 Monkey2SaccadeController Class Reference

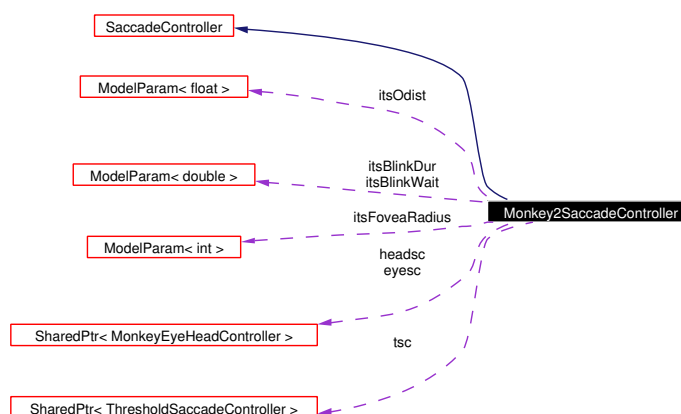
Realistic rhesus monkey saccade controller.

```
#include <SaccadeControllers.H>
```

Inheritance diagram for Monkey2SaccadeController:



Collaboration diagram for Monkey2SaccadeController:



### Public Member Functions

- [Monkey2SaccadeController](#) ([ModelManager](#) &mgr)

*Constructor.*

- virtual [~Monkey2SaccadeController](#) ()

*Destructor.*

- virtual void [exportOptions](#) (const int whatToExport, const bool recurse=true)

*Export some of our [ModelParam](#) parameters as command-line-options.*

- virtual void [evolve](#) (const double t)  
*evolve one time step (called, e.g., by [Brain::evolve\(\)](#))*

## Protected Member Functions

- virtual void [computeWhenNewPercept](#) ()  
*This method is called each time a new percept has arrived.*
- virtual void [computeWhenResetPos](#) ()  
*This method is called each time a reset is made.*
- virtual [Point2D](#) [computeWhenNewDecision](#) (const double t)  
*This method is called each time a new decision is requested.*
- virtual [Point2D](#) [computeWhenNewDecisionHead](#) (const double t)  
*This method is called each time a new head decision is requested.*

## Protected Attributes

- [ModelParam](#)< double > [itsBlinkWait](#)  
*wait time before a blink*
- [ModelParam](#)< double > [itsBlinkDur](#)  
*blink duration*
- [ModelParam](#)< float > [itsOdist](#)
- [ModelParam](#)< int > [itsFoveaRadius](#)

### 10.171.1 Detailed Description

Realistic rhesus monkey saccade controller.

This is a hack based on Nitin's [MonkeySaccadeController](#) but decoupling eye and head (so that the head minds its own business without influencing the eye). Not as realistic in the details, but the end result is the same

Definition at line 495 of file SaccadeControllers.H.

### 10.171.2 Constructor & Destructor Documentation

#### 10.171.2.1 **Monkey2SaccadeController::Monkey2SaccadeController** ([ModelManager](#) & *mgr*)

Constructor.

Definition at line 925 of file SaccadeControllers.C.

References [ModelComponent::addSubComponent\(\)](#), and [ModelManager::setOption-ValString\(\)](#).

#### 10.171.2.2 **Monkey2SaccadeController::~~Monkey2SaccadeController** () [virtual]

Destructor.

Definition at line 949 of file SaccadeControllers.C.

### 10.171.3 Member Function Documentation

#### 10.171.3.1 **Point2D** **Monkey2SaccadeController::computeWhenNewDecision** (const double *t*) [protected, virtual]

This method is called each time a new decision is requested.

Implements [SaccadeController](#).

Definition at line 1118 of file SaccadeControllers.C.

References [Point2D::i](#), and [Point2D::j](#).

#### 10.171.3.2 **Point2D** **Monkey2SaccadeController::computeWhen-** **NewDecisionHead** (const double *t*) [protected, virtual]

This method is called each time a new head decision is requested.

Reimplemented from [SaccadeController](#).

Definition at line 1138 of file SaccadeControllers.C.

References [Point2D::i](#), and [Point2D::j](#).

### 10.171.3.3 **void Monkey2SaccadeController::computeWhenNewPercept ()** [protected, virtual]

This method is called each time a new percept has arrived.

Implements [SaccadeController](#).

Definition at line 1025 of file SaccadeControllers.C.

References [SaccadeController::beginSaccade\(\)](#), [SaccadeController::beginSaccadeHead\(\)](#), [SaccadeController::getPreviousPercept\(\)](#), [ModelParam< float >::getVal\(\)](#), [ModelParam< int >::getVal\(\)](#), [Point2D::i](#), [SaccadeController::isInBlink\(\)](#), [Point2D::j](#), [WTAWinner::p](#), and [WTAWinner::t](#).

### 10.171.3.4 **void Monkey2SaccadeController::computeWhenResetPos ()** [protected, virtual]

This method is called each time a reset is made.

Implements [SaccadeController](#).

Definition at line 1112 of file SaccadeControllers.C.

### 10.171.3.5 **void Monkey2SaccadeController::evolve (const double t)** [virtual]

evolve one time step (called, e.g., by [Brain::evolve\(\)](#))

Implements [SaccadeController](#).

Definition at line 969 of file SaccadeControllers.C.

References [SaccadeController::beginBlink\(\)](#), [SaccadeController::endBlink\(\)](#), [SaccadeController::endSaccade\(\)](#), [SaccadeController::endSaccadeHead\(\)](#), [ModelParam< double >::getVal\(\)](#), [SaccadeController::isInBlink\(\)](#), [SaccadeController::isInSaccade\(\)](#), [itsBlinkDur](#), and [itsBlinkWait](#).

### 10.171.3.6 **void Monkey2SaccadeController::exportOptions (const int whatToExport, const bool recurse = true) [virtual]**

Export some of our [ModelParam](#) parameters as command-line-options.

See the base function in [ModelComponent.H](#) for info.

Reimplemented from [ModelComponent](#).

Definition at line 953 of file SaccadeControllers.C.

References [ModelComponent::exportOptions\(\)](#), [itsBlinkDur](#), [itsBlinkWait](#), [OPTEXP\\_CORE](#), and [ModelManager::requestOption\(\)](#).

## 10.171.4 Member Data Documentation

### 10.171.4.1 [ModelParam<double>](#) [Monkey2SaccadeController::itsBlinkDur](#) [protected]

blink duration

Definition at line 526 of file SaccadeControllers.H.

Referenced by [evolve\(\)](#), and [exportOptions\(\)](#).

### 10.171.4.2 [ModelParam<double>](#) [Monkey2SaccadeController::itsBlinkWait](#) [protected]

wait time before a blink

Definition at line 525 of file SaccadeControllers.H.

Referenced by [evolve\(\)](#), and [exportOptions\(\)](#).

The documentation for this class was generated from the following files:

- [SaccadeControllers.H](#)
- [SaccadeControllers.C](#)

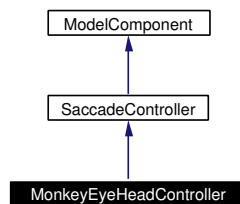


## 10.172 MonkeyEyeHeadController Class Reference

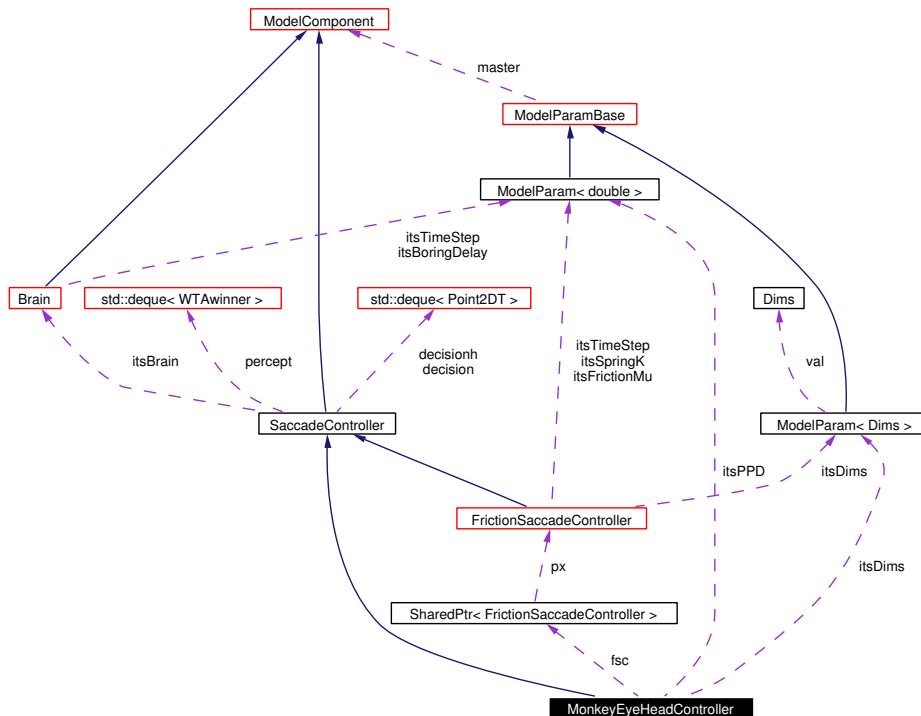
Realistic Monkey Eye/Head controller.

```
#include <SaccadeControllers.H>
```

Inheritance diagram for MonkeyEyeHeadController:



Collaboration diagram for MonkeyEyeHeadController:



## Public Member Functions

- [MonkeyEyeHeadController](#) ([ModelManager](#) &mgr, const [std::string](#) paramprefix="SCeye")  
*Constructor.*
- [~MonkeyEyeHeadController](#) ()  
*Destructor.*
- virtual void [exportOptions](#) (const int whatToExport, const bool recurse=true)  
*Export some of our [ModelParam](#) parameters as command-line-options.*
- void [saccade](#) (const [Point2DT](#) &target)  
*Initiate a realistic saccade.*
- void [pix2deg](#) (const [Point2D](#) &pixloc, double &xdeg, double &ydeg) const  
*Helper function converting from pixels to degrees.*
- void [deg2pix](#) (const double xdeg, const double ydeg, [Point2D](#) &pixloc) const  
*Helper function converting from degrees to pixels.*
- virtual void [evolve](#) (const double t)  
*evolve one time step (called, e.g., by [Brain::evolve\(\)](#))*

## Protected Member Functions

- virtual void [computeWhenNewPercept](#) ()  
*This method is called each time a new percept has arrived.*
- virtual void [computeWhenResetPos](#) ()  
*This method is called each time a reset is made.*
- virtual [Point2D](#) [computeWhenNewDecision](#) (const double t)  
*This method is called each time a new decision is requested.*

## Protected Attributes

- [ModelParam](#)< double > [itsPPD](#)  
*pixels per degree*

- [ModelParam](#) < [Dims](#) > [itsDims](#)

*Image* size.

### 10.172.1 Detailed Description

Realistic Monkey Eye/Head controller.

This is part of an attempt at a simplified approach to the [MonkeySaccadeController](#), which turned out quite complex because of the interdependency between head and eye movements. Here, we fake it by having independent controllers for the eyes and the head, such that we don't need to even deal with gaze stabilization during head movements and VOR. This controller will do either the head or the eye. For a controller that does both, see [Monkey2SaccadeController](#).

Definition at line 428 of file [SaccadeControllers.H](#).

### 10.172.2 Constructor & Destructor Documentation

#### 10.172.2.1 [MonkeyEyeHeadController::MonkeyEyeHeadController](#) ([ModelManager](#) & *mgr*, const [std::string](#) *paramprefix* = "SCeye")

Constructor.

##### Parameters:

*paramprefix* As this controller may be used either for the eye or for the head (as part of bigger controllers that include both eye and head), this should be either "SCeye" or "SChead" and will determine which model options we use for our internal parameters (eye or head)

Definition at line 747 of file [SaccadeControllers.C](#).

References [ModelComponent::addSubComponent\(\)](#).

#### 10.172.2.2 [MonkeyEyeHeadController::~~MonkeyEyeHeadController](#) ()

Destructor.

Definition at line 761 of file [SaccadeControllers.C](#).

### 10.172.3 Member Function Documentation

#### 10.172.3.1 **Point2D** MonkeyEyeHeadController::computeWhenNewDecision (const double t) [protected, virtual]

This method is called each time a new decision is requested.

Implements [SaccadeController](#).

Definition at line 912 of file SaccadeControllers.C.

References [SaccadeController::getPreviousDecision\(\)](#), [Point2D::i](#), and [Point2DT::p](#).

#### 10.172.3.2 **void** MonkeyEyeHeadController::computeWhenNewPercept () [protected, virtual]

This method is called each time a new percept has arrived.

Implements [SaccadeController](#).

Definition at line 848 of file SaccadeControllers.C.

References [SaccadeController::getPreviousPercept\(\)](#).

#### 10.172.3.3 **void** MonkeyEyeHeadController::computeWhenResetPos () [protected, virtual]

This method is called each time a reset is made.

Implements [SaccadeController](#).

Definition at line 898 of file SaccadeControllers.C.

References [SaccadeController::endSaccade\(\)](#), [SaccadeController::getPreviousPercept\(\)](#), [SaccadeController::isInSaccade\(\)](#), [WTAwinner::p](#), and [WTAwinner::t](#).

#### 10.172.3.4 **void** MonkeyEyeHeadController::deg2pix (const double xdeg, const double ydeg, **Point2D** & pixloc) const

Helper function converting from degrees to pixels.

Definition at line 834 of file SaccadeControllers.C.

References [itsPPD](#).

Referenced by [evolve\(\)](#).

**10.172.3.5 void MonkeyEyeHeadController::evolve (const double t)  
[virtual]**

evolve one time step (called, e.g., by [Brain::evolve\(\)](#))

Implements [SaccadeController](#).

Definition at line 858 of file SaccadeControllers.C.

References [deg2pix\(\)](#), [Point2D::i](#), [SaccadeController::isInSaccade\(\)](#), and [Point2D::j](#).

**10.172.3.6 void MonkeyEyeHeadController::exportOptions (const int  
whatToExport, const bool recurse = true) [virtual]**

Export some of our [ModelParam](#) parameters as command-line-options.

See the base function in [ModelComponent.H](#) for info.

Reimplemented from [ModelComponent](#).

Definition at line 765 of file SaccadeControllers.C.

References [ModelComponent::exportOptions\(\)](#), [itsPPD](#), [OPTEXP\\_CORE](#), and [ModelManager::requestOption\(\)](#).

**10.172.3.7 void MonkeyEyeHeadController::pix2deg (const [Point2D](#) & pixloc,  
double & xdeg, double & ydeg) const**

Helper function converting from pixels to degrees.

Definition at line 820 of file SaccadeControllers.C.

References [ModelParam< double >::getVal\(\)](#), [ModelParam< Dims >::getVal\(\)](#), [Dims::h\(\)](#), [Point2D::i](#), [itsPPD](#), [Point2D::j](#), and [Dims::w\(\)](#).

Referenced by [saccade\(\)](#).

**10.172.3.8 void MonkeyEyeHeadController::saccade (const [Point2DT](#) & target)**

Initiate a realistic saccade.

Definition at line 779 of file SaccadeControllers.C.

References [SaccadeController::beginSaccade\(\)](#), [exp\(\)](#), [pix2deg\(\)](#), and [sqrt\(\)](#).

## 10.172.4 Member Data Documentation

### 10.172.4.1 [ModelParam<Dims>](#) [MonkeyEyeHeadController::itsDims](#) [protected]

[Image](#) size.

Definition at line 475 of file SaccadeControllers.H.

### 10.172.4.2 [ModelParam<double>](#) [MonkeyEyeHeadController::itsPPD](#) [protected]

pixels per degree

Definition at line 474 of file SaccadeControllers.H.

Referenced by [deg2pix\(\)](#), [exportOptions\(\)](#), and [pix2deg\(\)](#).

The documentation for this class was generated from the following files:

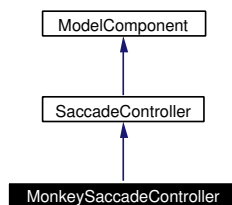
- [SaccadeControllers.H](#)
- [SaccadeControllers.C](#)

## 10.173 MonkeySaccadeController Class Reference

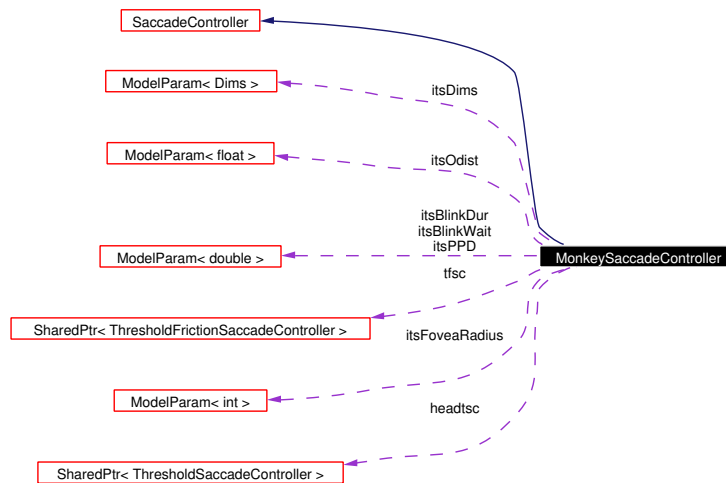
Realistic rhesus monkey saccade controller.

```
#include <SaccadeControllers.H>
```

Inheritance diagram for MonkeySaccadeController:



Collaboration diagram for MonkeySaccadeController:



### Public Member Functions

- [MonkeySaccadeController](#) ([ModelManager](#) &mgr)  
*Constructor.*
- virtual [~MonkeySaccadeController](#) ()  
*Destructor.*

- virtual void [exportOptions](#) (const int whatToExport, const bool recurse=true)  
*Export some of our [ModelParam](#) parameters as command-line-options.*
- virtual void [evolve](#) (const double t)  
*evolve one time step (called, e.g., by [Brain::evolve\(\)](#))*
- [Point2DT](#) [getEye](#) () const  
*Get the current position of the eyes (same as returned decisions).*
- [Point2DT](#) [getHead](#) () const  
*Get the current direction of the head.*
- [Point2DT](#) [getJustEye](#) () const  
*Get the current direction of the eye.*

### Protected Member Functions

- virtual void [computeWhenNewPercept](#) ()  
*This method is called each time a new percept has arrived.*
- virtual void [computeWhenResetPos](#) ()  
*This method is called each time a reset is made.*
- virtual [Point2D](#) [computeWhenNewDecision](#) (const double t)  
*This method is called each time a new decision is requested.*

### Protected Attributes

- [ModelParam](#)< double > [itsBlinkWait](#)  
*wait time before a blink*
- [ModelParam](#)< double > [itsBlinkDur](#)  
*blink duration*
- [ModelParam](#)< float > [itsOdist](#)
- [ModelParam](#)< int > [itsFoveaRadius](#)
- [ModelParam](#)< double > [itsPPD](#)  
*pixels per degree*
- [ModelParam](#)< [Dims](#) > [itsDims](#)



*Image* size.

- [SharedPtr](#)< [ThresholdSaccadeController](#) > **headtsc**
- [SharedPtr](#)< [ThresholdFrictionSaccadeController](#) > **tfsc**

### 10.173.1 Detailed Description

Realistic rhesus monkey saccade controller.

Nitin's masterpiece!

Definition at line 545 of file SaccadeControllers.H.

### 10.173.2 Constructor & Destructor Documentation

#### 10.173.2.1 [MonkeySaccadeController::MonkeySaccadeController](#) ([ModelManager](#) & *mgr*)

Constructor.

Definition at line 1199 of file SaccadeControllers.C.

References [ModelComponent::addSubComponent\(\)](#), and [eye\(\)](#).

#### 10.173.2.2 [MonkeySaccadeController::~~MonkeySaccadeController](#) () [virtual]

Destructor.

Definition at line 1221 of file SaccadeControllers.C.

### 10.173.3 Member Function Documentation

#### 10.173.3.1 [Point2D](#) [MonkeySaccadeController::computeWhenNewDecision](#) (**const double** *t*) [protected, virtual]

This method is called each time a new decision is requested.

Implements [SaccadeController](#).

Definition at line 1301 of file SaccadeControllers.C.

References [SaccadeController::endSaccade\(\)](#), [exp\(\)](#), [eye\(\)](#), [getEye\(\)](#), [SaccadeController::getPreviousPercept\(\)](#), [ModelParam< float >::getVal\(\)](#), [ModelParam< int >::getVal\(\)](#), [ModelParam< double >::getVal\(\)](#), [ModelParam< Dims >::getVal\(\)](#), [Dims::h\(\)](#), [SaccadeController::havePercepts\(\)](#), [Point2D::i](#), [Point2D::j](#), [WTAwinner::p](#), [Point2DT::p](#), [sqrt\(\)](#), [WTAwinner::t](#), and [Dims::w\(\)](#).

**10.173.3.2 void MonkeySaccadeController::computeWhenNewPercept ()**  
[protected, virtual]

This method is called each time a new percept has arrived.

Implements [SaccadeController](#).

Definition at line 1280 of file SaccadeControllers.C.

References [SaccadeController::getPreviousPercept\(\)](#).

**10.173.3.3 void MonkeySaccadeController::computeWhenResetPos ()**  
[protected, virtual]

This method is called each time a reset is made.

Implements [SaccadeController](#).

Definition at line 1290 of file SaccadeControllers.C.

References [SaccadeController::getPreviousPercept\(\)](#), [WTAwinner::p](#), and [WTAwinner::t](#).

**10.173.3.4 void MonkeySaccadeController::evolve (const double t)**  
[virtual]

evolve one time step (called, e.g., by [Brain::evolve\(\)](#))

Implements [SaccadeController](#).

Definition at line 1243 of file SaccadeControllers.C.

References [SaccadeController::beginBlink\(\)](#), [SaccadeController::endBlink\(\)](#), [ModelParam< double >::getVal\(\)](#), and [SaccadeController::isInBlink\(\)](#).

**10.173.3.5 void MonkeySaccadeController::exportOptions (const int  
whatToExport, const bool recurse = true) [virtual]**

Export some of our [ModelParam](#) parameters as command-line-options.

See the base function in [ModelComponent.H](#) for info.

Reimplemented from [ModelComponent](#).

Definition at line 1225 of file SaccadeControllers.C.

References [ModelComponent::exportOptions\(\)](#), [OPTEXP\\_CORE](#), and [ModelManager::requestOption\(\)](#).

### 10.173.3.6 [Point2DT](#) MonkeySaccadeController::getEye () const

Get the current position of the eyes (same as returned decisions).

Definition at line 1615 of file SaccadeControllers.C.

References [ModelParam< double >::getVal\(\)](#), [ModelParam< Dims >::getVal\(\)](#), [Dims::h\(\)](#), [Point2D::i](#), [Point2D::j](#), [Point2DT::p](#), [Point2DT::t](#), and [Dims::w\(\)](#).

Referenced by [computeWhenNewDecision\(\)](#).

### 10.173.3.7 [Point2DT](#) MonkeySaccadeController::getHead () const

Get the current direction of the head.

Definition at line 1645 of file SaccadeControllers.C.

References [ModelParam< double >::getVal\(\)](#), [ModelParam< Dims >::getVal\(\)](#), [Dims::h\(\)](#), [Point2D::i](#), [Point2D::j](#), [Point2DT::p](#), [Point2DT::t](#), and [Dims::w\(\)](#).

### 10.173.3.8 [Point2DT](#) MonkeySaccadeController::getJustEye () const

Get the current direction of the eye.

Definition at line 1630 of file SaccadeControllers.C.

References [ModelParam< double >::getVal\(\)](#), [ModelParam< Dims >::getVal\(\)](#), [Dims::h\(\)](#), [Point2D::i](#), [Point2D::j](#), [Point2DT::p](#), [Point2DT::t](#), and [Dims::w\(\)](#).

## 10.173.4 Member Data Documentation

### 10.173.4.1 [ModelParam<double>](#) [MonkeySaccadeController::itsBlinkDur](#) [protected]

blink duration

Definition at line 582 of file SaccadeControllers.H.

### 10.173.4.2 [ModelParam<double>](#) [MonkeySaccadeController::itsBlinkWait](#) [protected]

wait time before a blink

Definition at line 581 of file SaccadeControllers.H.

#### 10.173.4.3 [ModelParam<Dims>](#) [MonkeySaccadeController::itsDims](#) [protected]

[Image](#) size.

Definition at line 586 of file SaccadeControllers.H.

#### 10.173.4.4 [ModelParam<double>](#) [MonkeySaccadeController::itsPPD](#) [protected]

pixels per degree

Definition at line 585 of file SaccadeControllers.H.

The documentation for this class was generated from the following files:

- [SaccadeControllers.H](#)
- [SaccadeControllers.C](#)



*(re-)build our subchannels*

- virtual void `doInput` (const `Image`< `PixRGB`< float > > \*, const `Image`< float > \*bwimg, const double t, const `Image`< byte > &clipMask)

*MotionChannel only required luminosity input.*

## Protected Attributes

- `ModelParam`< `PyramidType` > `itsPyrType`  
*type of pyramid to use in our `DirectionChannel` subcomponents*
- `ModelParam`< `uint` > `itsNumDirs`  
*number of `DirectionChannel` subcomponents*
- int `itsExportOpts`  
*keep track of what options to export in case we rebuild our subchans*

### 10.174.1 Detailed Description

A composite channel containing a set of direction channels.

Definition at line 1145 of file `Channels.H`.

### 10.174.2 Constructor & Destructor Documentation

#### 10.174.2.1 `MotionChannel::MotionChannel` (`ModelManager` & *mgr*)

Construct.

Definition at line 1463 of file `Channels.C`.

References `buildSubChans()`, `Gaussian5`, and `OPTEXP_NONE`.

#### 10.174.2.2 `MotionChannel::~~MotionChannel` () [virtual]

destructor

Definition at line 1475 of file `Channels.C`.

### 10.174.3 Member Function Documentation

#### 10.174.3.1 `void MotionChannel::buildSubChans ()` [protected, virtual]

(re-)build our subchannels

Reimplemented in [MotionChannelBeo](#), and [MotionChannelSurprise](#).

Definition at line 1497 of file Channels.C.

References [ComplexChannel::addSubChan\(\)](#), [ModelComponent::exportOptions\(\)](#), [ModelParam< PyramidType >::getVal\(\)](#), [ModelParam< uint >::getVal\(\)](#), [itsNumDirs](#), [itsPyrType](#), [makeShared\(\)](#), [OPTEXP\\_NONE](#), [ModelComponent::removeAllSubComponents\(\)](#), [ChannelBase::setHomogeneous\(\)](#), and [uint](#).

Referenced by [MotionChannel\(\)](#), and [setModelParamString\(\)](#).

#### 10.174.3.2 `DirectionChannel & MotionChannel::dirChan (const uint idx) const` [virtual]

returns a specific [DirectionChannel](#)

Reimplemented in [MotionChannelSurprise](#).

Definition at line 1493 of file Channels.C.

References [ComplexChannel::subChan\(\)](#), and [uint](#).

#### 10.174.3.3 `void MotionChannel::doInput (const Image< PixRGB< float > > *, const Image< float > * bwimg, const double t, const Image< byte > & clipMask)` [protected, virtual]

MotionChannel only required luminosity input.

Implements [ChannelBase](#).

Definition at line 1540 of file Channels.C.

References [ASSERT](#), [ChannelBase::input\(\)](#), [ComplexChannel::numChans\(\)](#), [ComplexChannel::subChan\(\)](#), and [uint](#).

#### 10.174.3.4 `void MotionChannel::exportOptions (const int whatToExport, const bool recurse = true)` [virtual]

Export some of our [ModelParam](#) parameters as command-line-options.

See the base function in [ModelComponent.H](#) for info.

Reimplemented from [ComplexChannel](#).

Definition at line 1479 of file Channels.C.

References `ComplexChannel::exportOptions()`, `itsNumDirs`, `OPTEXP_CORE`, and `ModelManager::requestOption()`.

**10.174.3.5** `bool MotionChannel::setModelParamString (const std::string & name, const std::string & value, const bool recurse = false, const bool warn = true) [virtual]`

Overload so that we can reconfigure when our params get changed.

Reimplemented from [ModelComponent](#).

Definition at line 1522 of file Channels.C.

References `buildSubChans()`, `ModelParamBase::getName()`, `ModelParam< uint >::getVal()`, `itsNumDirs`, `ComplexChannel::numChans()`, and `ModelComponent::setModelParamString()`.

## 10.174.4 Member Data Documentation

**10.174.4.1** `int MotionChannel::itsExportOpts [protected]`

keep track of what options to export in case we rebuild our subchans

Definition at line 1184 of file Channels.H.

**10.174.4.2** `ModelParam<uint> MotionChannel::itsNumDirs [protected]`

number of [DirectionChannel](#) subcomponents

Definition at line 1173 of file Channels.H.

Referenced by `buildSubChans()`, `exportOptions()`, and `setModelParamString()`.

**10.174.4.3** `ModelParam<PyramidType> MotionChannel::itsPyrType [protected]`

type of pyramid to use in our [DirectionChannel](#) subcomponents

Definition at line 1170 of file Channels.H.

Referenced by `buildSubChans()`.

The documentation for this class was generated from the following files:

- [Channels.H](#)
- [Channels.C](#)

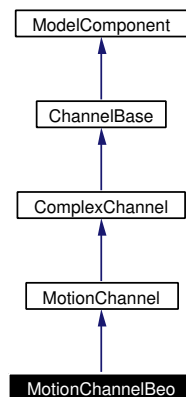


## 10.175 MotionChannelBeo Class Reference

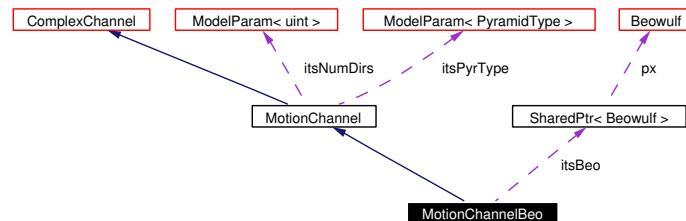
Beowulf version of [MotionChannel](#).

```
#include <ChannelsBeo.H>
```

Inheritance diagram for MotionChannelBeo:



Collaboration diagram for MotionChannelBeo:



### Public Member Functions

- **MotionChannelBeo** ([ModelManager](#) &mgr, const [SharedPtr< Beowulf >](#) &beow)

### Protected Member Functions

- virtual void [buildSubChans](#) ()  
(re-)build our subchannels

### 10.175.1 Detailed Description

[Beowulf](#) version of [MotionChannel](#).

Definition at line 202 of file ChannelsBeo.H.

### 10.175.2 Member Function Documentation

#### 10.175.2.1 `void MotionChannelBeo::buildSubChans ()` [protected, virtual]

(re-)build our subchannels

Reimplemented from [MotionChannel](#).

Definition at line 322 of file ChannelsBeo.C.

References [ComplexChannel::addSubChan\(\)](#), [ModelComponent::exportOptions\(\)](#), [ModelParam< PyramidType >::getVal\(\)](#), [ModelParam< uint >::getVal\(\)](#), [OPTEXP\\_-NONE](#), [ModelComponent::removeAllSubComponents\(\)](#), [ChannelBase::setHomogeneous\(\)](#), and [uint](#).

The documentation for this class was generated from the following files:

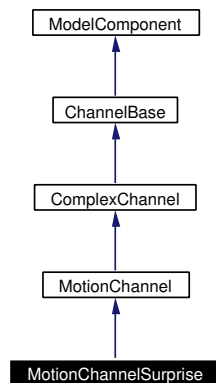
- [ChannelsBeo.H](#)
- [ChannelsBeo.C](#)

## 10.176 MotionChannelSurprise Class Reference

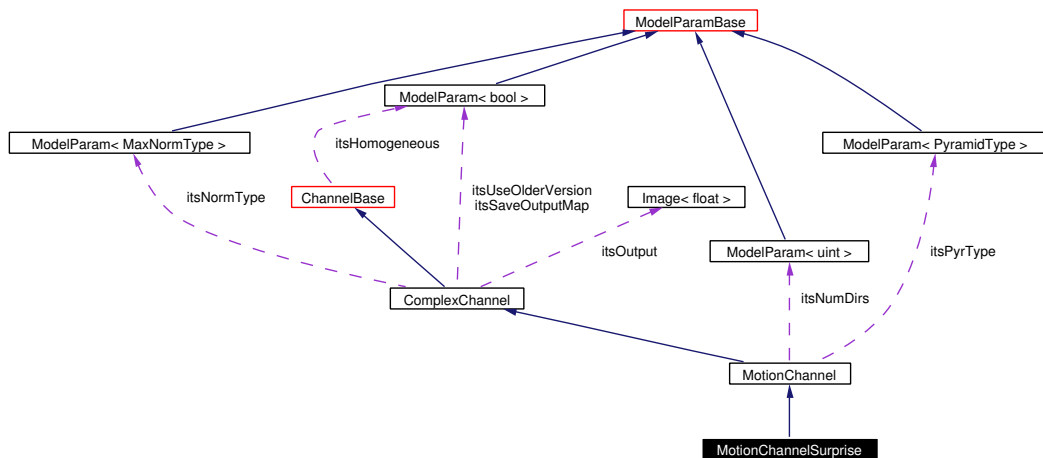
A composite channel containing a set of direction channels.

```
#include <ChannelsSurprise.H>
```

Inheritance diagram for MotionChannelSurprise:



Collaboration diagram for MotionChannelSurprise:



### Public Member Functions

- [MotionChannelSurprise](#) ([ModelManager](#) &mgr)

*Construct.*

- virtual [~MotionChannelSurprise](#) ()

*destructor*

- virtual [DirectionChannelSurprise](#) & [dirChan](#) (const [uint](#) idx) const

*returns a specific [DirectionChannelSurprise](#)*

## Protected Member Functions

- virtual void [buildSubChans](#) ()

*(re-)build our subchannels*

### 10.176.1 Detailed Description

A composite channel containing a set of direction channels.

Definition at line 545 of file ChannelsSurprise.H.

### 10.176.2 Constructor & Destructor Documentation

#### 10.176.2.1 [MotionChannelSurprise::MotionChannelSurprise](#) ([ModelManager](#) & *mgr*)

Construct.

Definition at line 632 of file ChannelsSurprise.C.

References [buildSubChans](#)().

#### 10.176.2.2 [MotionChannelSurprise::~MotionChannelSurprise](#) () [virtual]

destructor

Definition at line 641 of file ChannelsSurprise.C.

### 10.176.3 Member Function Documentation

#### 10.176.3.1 `void MotionChannelSurprise::buildSubChans ()` [protected, virtual]

(re-)build our subchannels

Reimplemented from [MotionChannel](#).

Definition at line 649 of file ChannelsSurprise.C.

References [ComplexChannel::addSubChan\(\)](#), [ModelComponent::exportOptions\(\)](#), [ModelParam< PyramidType >::getVal\(\)](#), [ModelParam< uint >::getVal\(\)](#), [makeShared\(\)](#), [OPTEXP\\_NONE](#), [ModelComponent::removeAllSubComponents\(\)](#), [ChannelBase::setHomogeneous\(\)](#), and [uint](#).

Referenced by [MotionChannelSurprise\(\)](#).

#### 10.176.3.2 `DirectionChannelSurprise & MotionChannelSurprise::dirChan (const uint idx) const` [virtual]

returns a specific [DirectionChannelSurprise](#)

Reimplemented from [MotionChannel](#).

Definition at line 645 of file ChannelsSurprise.C.

References [ComplexChannel::subChan\(\)](#), and [uint](#).

The documentation for this class was generated from the following files:

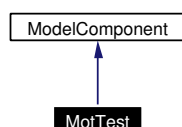
- [ChannelsSurprise.H](#)
- [ChannelsSurprise.C](#)

## 10.177 MotTest Class Reference

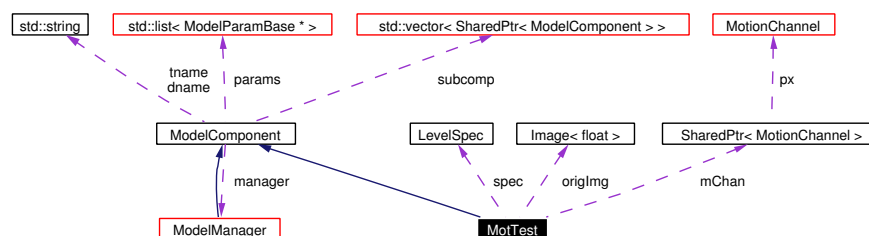
Class for testing [MotionChannel](#).

```
#include <MotTest.H>
```

Inheritance diagram for MotTest:



Collaboration diagram for MotTest:



### Public Types

- enum **enumSM** { STAT, MAP }
- enum **enumGDM** { GABOR, DIRECTION, MOTION }
- enum **enumPFC** { PYRAMID, FEATURE, CONSPICUITY }

### Public Member Functions

- **MotTest** ([ModelManager](#) &mgr, bool useIntensity)
- [Image< byte >](#) **loadImage** ([std::string](#) fName, [RasterFileFormat](#) ftype)
- void **inputChan** ([Image< byte >](#) img)
- void **dispMinMaxAvg** ([Image< float >](#) img)
- void **dispImage** ([Image< float >](#) img, [std::string](#) caption)
- void **normalize** (bool bNorm)
- void **writeSixPack** (int frmNum)
- [Image< float >](#) **getImage** (enumGDM eGDM, int idxGDM, enumPFC ePFC, int levPFC)

- void **display** (enumSM eSM, enumGDM eGDM, int idxGDM, enumPFC ePFC, int levPFC)

### 10.177.1 Detailed Description

Class for testing [MotionChannel](#).

This is a test class providing a nice interface for testing the [MotionChannel](#). It constructs the [MotionChannel](#) in its constructor and provides methods to load images, input the images, display the output maps, display statistics for the maps, etc.

The most useful function probably is dispSixPack(int) which takes the current frame number as input and writes the outputs of the four DirectionChannels, the [MotionChannel](#), and the original image to a file named Six.XXXXX.ppm - which can then be fed through mq for quicktime movie creation.

Definition at line 74 of file MotTest.H.

The documentation for this class was generated from the following files:

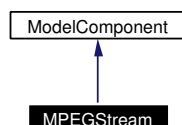
- [MotTest.H](#)
- [MotTest.C](#)

## 10.178 MPEGStream Class Reference

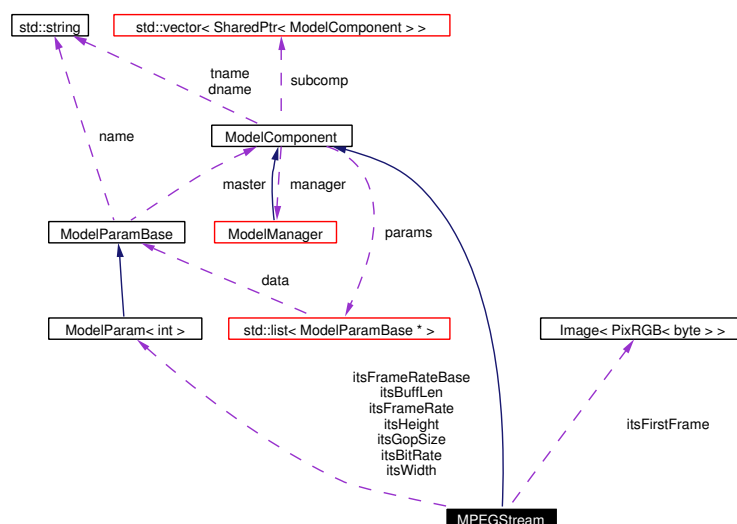
MPEGStream is a wrapper class to read video frames from mpeg-1.

```
#include <MPEGStream.H>
```

Inheritance diagram for MPEGStream:



Collaboration diagram for MPEGStream:



### Public Member Functions

- **MPEGStream** (**ModelManager** &mgr, const **std::string** &descrName="MPEG Stream Abstraction", const **std::string** &tagName="MPEGStreamer")

*Construct an mpegstream object.*

- virtual void **exportOptions** (const int whatToExport, const bool recurse=true)

*Export some of our **ModelParam** parameters as command-line-options.*



- virtual `~MPEGStream ()`  
*virtual destructor since we inherit from [ModelComponent](#)*
- int `initDecode (std::string fname, const bool preload=false)`  
*Configure the MPEG Stream for decoding.*
- int `initEncode (std::string fname, const int width=0, const int height=0)`  
*Configure the MPEG Stream for encoding.*
- `Image< PixRGB< byte > > ReadFrame ()`  
*Call this to get the next frame from the stream.*
- const AVFrame \* `ReadFrameRaw ()`  
*Call this to get the next raw YV12 frame from the stream.*
- void `WriteFrame (Image< PixRGB< byte > > img)`  
*write a frame to the stream*
- int `getWidth ()`  
*get frame width*
- int `getHeight ()`  
*get frame height*

## Protected Member Functions

- void `stop2 ()`  
*get stopped; close any open write stream*

## Protected Attributes

- `ModelParam< int > itsBitRate`
- `ModelParam< int > itsWidth`
- `ModelParam< int > itsHeight`
- `ModelParam< int > itsFrameRate`
- `ModelParam< int > itsFrameRateBase`
- `ModelParam< int > itsGopSize`
- `ModelParam< int > itsBuffLen`
- `ModelParam< std::string > itsCodecName`

### 10.178.1 Detailed Description

MPEGStream is a wrapper class to read video frames from mpeg-1.

MPEGStream is a wrapper class to read video frames from mpeg-1, mpeg-4 (aka divx), asf, avi, mov and any other format that mplayer can decode. Audio streams are entirely ignored.

Definition at line 64 of file MPEGStream.H.

### 10.178.2 Constructor & Destructor Documentation

#### 10.178.2.1 MPEGStream::MPEGStream ([ModelManager](#) & mgr, const [std::string](#) & descrName = "MPEG Stream Abstraction", const [std::string](#) & tagName = "MPEGStreamer")

Construct an mpegstream object.

FIXME: Do not forget to the set the height and width of the decoded frame (see the parameters itsWidth & itsHeight)

Definition at line 311 of file MPEGStream.C.

#### 10.178.2.2 MPEGStream::~~MPEGStream () [virtual]

virtual destructor since we inherit from [ModelComponent](#)

Definition at line 362 of file MPEGStream.C.

### 10.178.3 Member Function Documentation

#### 10.178.3.1 void MPEGStream::exportOptions (const int *whatToExport*, const bool *recurse* = true) [virtual]

Export some of our [ModelParam](#) parameters as command-line-options.

See the base function in [ModelComponent.H](#) for info.

Reimplemented from [ModelComponent](#).

Definition at line 328 of file MPEGStream.C.

References [ModelComponent::exportOptions\(\)](#), `OPTEXP_CORE`, and [ModelManager::requestOption\(\)](#).

**10.178.3.2 int MPEGStream::getHeight ()**

get frame height

Definition at line 544 of file MPEGStream.C.

References ReadFrame().

**10.178.3.3 int MPEGStream::getWidth ()**

get frame width

Definition at line 533 of file MPEGStream.C.

References ReadFrame().

**10.178.3.4 int MPEGStream::initDecode (std::string fname, const bool preload = false)**

Configure the MPEG Stream for decoding.

**Parameters:**

**fname** name of file from which frames are to be read from

**preload** preload the entire movie into memory if true.

Definition at line 366 of file MPEGStream.C.

References Image< PixRGB< byte > >::freeMem(), and ModelParam< int >::getVal().

**10.178.3.5 int MPEGStream::initEncode (std::string fname, const int width = 0, const int height = 0)**

Configure the MPEG Stream for encoding.

if width and height are specified (non-zero), they will override the [ModelParam](#) values of itsWidth and itsHeight

Definition at line 455 of file MPEGStream.C.

References ModelParam< int >::getVal(), and uint.

**10.178.3.6 Image< PixRGB< byte > > MPEGStream::ReadFrame ()**

Call this to get the next frame from the stream.

returns an image of size (0,0) at the end of the stream

Definition at line 555 of file MPEGStream.C.

References [Image< PixRGB< byte > >::freeMem\(\)](#), [Image< PixRGB< byte > >::initialized\(\)](#), and [ReadFrameRaw\(\)](#).

Referenced by [getHeight\(\)](#), and [getWidth\(\)](#).

#### 10.178.3.7 **const AVFrame \* MPEGStream::ReadFrameRaw ()**

Call this to get the next raw YV12 frame from the stream.

returns NULL if reached end of stream. The buffer returned is allocated internally, so you should not attempt to free it. The next call to [ReadFrame\(\)](#) or [ReadFrameRaw\(\)](#) will overwrite the frame.

Definition at line 579 of file MPEGStream.C.

References [ModelParam< int >::getVal\(\)](#).

Referenced by [ReadFrame\(\)](#).

#### 10.178.3.8 **void MPEGStream::stop2 ()** [protected, virtual]

get stopped; close any open write stream

Reimplemented from [ModelComponent](#).

Definition at line 339 of file MPEGStream.C.

#### 10.178.3.9 **void MPEGStream::WriteFrame (Image< PixRGB< byte > > img)**

write a frame to the stream

Definition at line 620 of file MPEGStream.C.

References [Image< T >::getArrayPtr\(\)](#), [Image< T >::getHeight\(\)](#), and [Image< T >::getWidth\(\)](#).

The documentation for this class was generated from the following files:

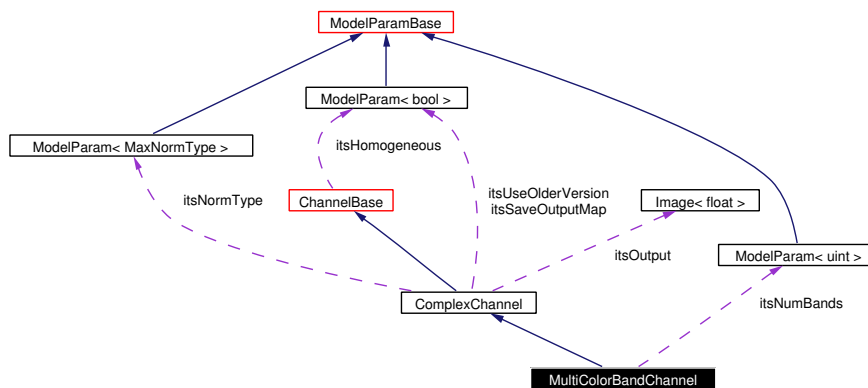
- [MPEGStream.H](#)
- [MPEGStream.C](#)

## 10.179 MultiColorBandChannel Class Reference

A composite channel containing a set of color-band channels.

```
#include <Channels.H>
```

Inheritance diagram for MultiColorBandChannel: Collaboration diagram for MultiColorBandChannel:



### Public Member Functions

- [MultiColorBandChannel](#) ([ModelManager](#) &mgr)  
*Construct with standard params.*
- virtual [~MultiColorBandChannel](#) ()  
*Destructor.*
- virtual void [exportOptions](#) (const int whatToExport, const bool recurse=true)  
*Export some of our [ModelParam](#) parameters as command-line-options.*
- virtual [ColorBandChannel](#) & [band](#) (const [uint](#) idx) const  
*Wraps [ComplexChannel](#)'s [subChan\(\)](#) and casts the result to [ColorBandChannel](#).*
- virtual bool [setModelParamString](#) (const [std::string](#) &name, const [std::string](#) &value, const bool recurse=false, const bool warn=true)  
*Overload so that we can reconfigure when our params get changed.*

## Protected Member Functions

- virtual void [buildSubChans](#) ()  
*(re-)build our subchannels*
- virtual void [doInput](#) (const [Image](#)< [PixRGB](#)< float > > \*colimg, const [Image](#)< float > \*, const double t, const [Image](#)< byte > &clipMask)  
*[ColorBandChannel](#) requires only the color image input.*

## Protected Attributes

- [ModelParam](#)< uint > [itsNumBands](#)  
*number of color bands*
- int [itsExportOpts](#)  
*keep track of what options to export in case we rebuild our subchans*

### 10.179.1 Detailed Description

A composite channel containing a set of color-band channels.

This is an example of a homogeneous complex channel. Thus, the [JetSpec](#) representation will have an index for the subchannel number, and further indices for the various [Jet](#) components (which are identical for all subchannels).

Definition at line 1104 of file Channels.H.

### 10.179.2 Constructor & Destructor Documentation

#### 10.179.2.1 [MultiColorBandChannel::MultiColorBandChannel](#) ([ModelManager](#) & mgr)

Construct with standard params.

Definition at line 1307 of file Channels.C.

References [buildSubChans\(\)](#), [COLBAND](#), and [OPTEXP\\_NONE](#).

#### 10.179.2.2 [MultiColorBandChannel::~~MultiColorBandChannel](#) () [virtual]

Destructor.

Definition at line 1358 of file Channels.C.

### 10.179.3 Member Function Documentation

#### 10.179.3.1 [ColorBandChannel](#) & [MultiColorBandChannel::band](#) (const [uint](#) *idx*) const [virtual]

Wraps [ComplexChannel](#)'s [subChan\(\)](#) and casts the result to [ColorBandChannel](#).

Reimplemented in [MultiColorBandChannelSurprise](#).

Definition at line 1377 of file Channels.C.

References [ComplexChannel::subChan\(\)](#), and [uint](#).

#### 10.179.3.2 void [MultiColorBandChannel::buildSubChans](#) () [protected, virtual]

(re-)build our subchannels

Reimplemented in [MultiColorBandChannelBeo](#), and [MultiColorBandChannelSurprise](#).

Definition at line 1318 of file Channels.C.

References [ComplexChannel::addSubChan\(\)](#), [ModelComponent::exportOptions\(\)](#), [ModelParam< uint >::getVal\(\)](#), [itsNumBands](#), [makeShared\(\)](#), [OPTEXP\\_NONE](#), [ModelComponent::removeAllSubComponents\(\)](#), [ChannelBase::setHomogeneous\(\)](#), and [uint](#).

Referenced by [MultiColorBandChannel\(\)](#), and [setModelParamString\(\)](#).

#### 10.179.3.3 void [MultiColorBandChannel::doInput](#) (const [Image](#)< [PixRGB](#)< [float](#) > > \* *colimg*, const [Image](#)< [float](#) > \*, const double *t*, const [Image](#)< [byte](#) > & *clipMask*) [protected, virtual]

[ColorBandChannel](#) requires only the color image input.

Implements [ChannelBase](#).

Definition at line 1385 of file Channels.C.

References [ASSERT](#), [getComponents\(\)](#), [ChannelBase::input\(\)](#), [ComplexChannel::numChans\(\)](#), and [ComplexChannel::subChan\(\)](#).

#### 10.179.3.4 `void MultiColorBandChannel::exportOptions (const int whatToExport, const bool recurse = true) [virtual]`

Export some of our [ModelParam](#) parameters as command-line-options.

See the base function in [ModelComponent.H](#) for info.

Reimplemented from [ComplexChannel](#).

Definition at line 1362 of file Channels.C.

References [ComplexChannel::exportOptions\(\)](#), [itsNumBands](#), [OPTEXP\\_CORE](#), and [ModelManager::requestOption\(\)](#).

#### 10.179.3.5 `bool MultiColorBandChannel::setModelParamString (const std::string & name, const std::string & value, const bool recurse = false, const bool warn = true) [virtual]`

Overload so that we can reconfigure when our params get changed.

Reimplemented from [ModelComponent](#).

Definition at line 1340 of file Channels.C.

References [buildSubChans\(\)](#), [ModelParamBase::getName\(\)](#), [ModelParam<uint >::getVal\(\)](#), [itsNumBands](#), [ComplexChannel::numChans\(\)](#), and [ModelComponent::setModelParamString\(\)](#).

### 10.179.4 Member Data Documentation

#### 10.179.4.1 `int MultiColorBandChannel::itsExportOpts [protected]`

keep track of what options to export in case we rebuild our subchans

Definition at line 1139 of file Channels.H.

#### 10.179.4.2 `ModelParam<uint> MultiColorBandChannel::itsNumBands [protected]`

number of color bands

Definition at line 1128 of file Channels.H.

Referenced by [buildSubChans\(\)](#), [exportOptions\(\)](#), and [setModelParamString\(\)](#).

The documentation for this class was generated from the following files:

- [Channels.H](#)
- [Channels.C](#)

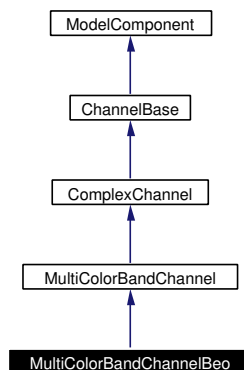


## 10.180 MultiColorBandChannelBeo Class Reference

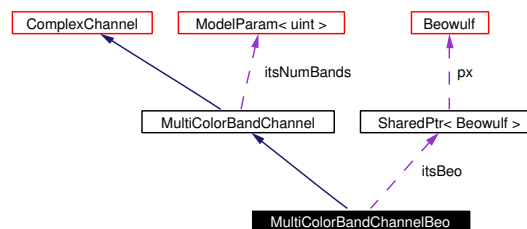
Beowulf version of [MultiColorBandChannel](#).

```
#include <ChannelsBeo.H>
```

Inheritance diagram for MultiColorBandChannelBeo:



Collaboration diagram for MultiColorBandChannelBeo:



### Public Member Functions

- **MultiColorBandChannelBeo** ([ModelManager](#) &mgr, const [SharedPtr< Beowulf >](#) &beow)

### Protected Member Functions

- virtual void [buildSubChans](#) ()  
(re-)build our subchannels

### 10.180.1 Detailed Description

Beowulf version of [MultiColorBandChannel](#).

Definition at line 189 of file ChannelsBeo.H.

### 10.180.2 Member Function Documentation

#### 10.180.2.1 `void MultiColorBandChannelBeo::buildSubChans ()` [protected, virtual]

(re-)build our subchannels

Reimplemented from [MultiColorBandChannel](#).

Definition at line 295 of file ChannelsBeo.C.

References [ComplexChannel::addSubChan\(\)](#), [ModelComponent::exportOptions\(\)](#), [ModelParam< uint >::getVal\(\)](#), [OPTEXP\\_NONE](#), [ModelComponent::removeAllSubComponents\(\)](#), [ChannelBase::setHomogeneous\(\)](#), and [uint](#).

The documentation for this class was generated from the following files:

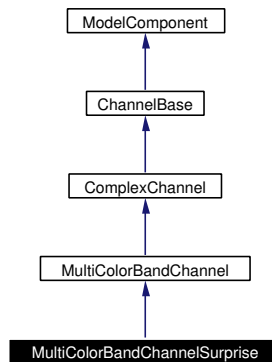
- [ChannelsBeo.H](#)
- [ChannelsBeo.C](#)

## 10.181 MultiColorBandChannelSurprise Class Reference

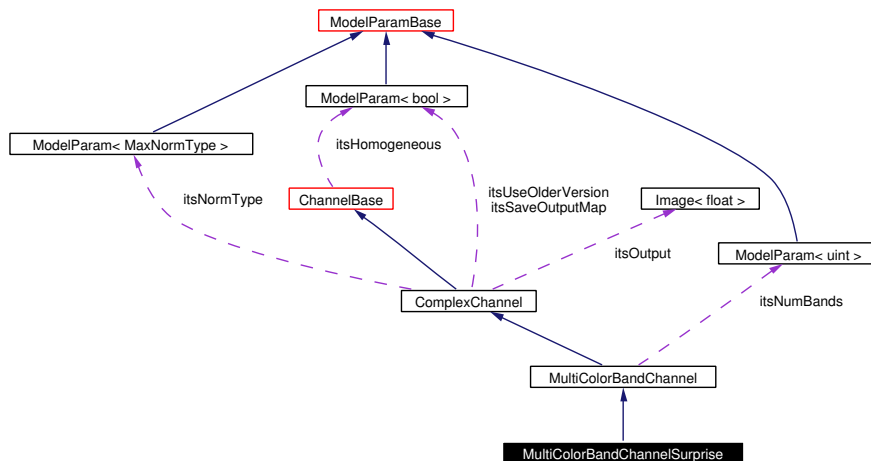
A composite channel containing a set of color-band channels.

```
#include <ChannelsSurprise.H>
```

Inheritance diagram for MultiColorBandChannelSurprise:



Collaboration diagram for MultiColorBandChannelSurprise:



### Public Member Functions

- [MultiColorBandChannelSurprise](#) (`ModelManager` &`mgr`)

*Construct with standard params.*

- virtual [~MultiColorBandChannelSurprise](#) ()

*Destructor.*

- virtual [ColorBandChannelSurprise](#) & [band](#) (const [uint](#) idx) const

*Wraps ComplexChannel's [subChan\(\)](#) and casts to [ColorBandChannelSurprise](#).*

## Protected Member Functions

- virtual void [buildSubChans](#) ()

*(re-)build our subchannels*

### 10.181.1 Detailed Description

A composite channel containing a set of color-band channels.

Definition at line 526 of file ChannelsSurprise.H.

### 10.181.2 Constructor & Destructor Documentation

#### 10.181.2.1 MultiColorBandChannelSurprise::MultiColor-BandChannelSurprise ([ModelManager](#) & *mgr*)

Construct with standard params.

Definition at line 587 of file ChannelsSurprise.C.

References [buildSubChans\(\)](#).

#### 10.181.2.2 MultiColorBandChannelSurprise::~~MultiColorBandChannel-Surprise () [virtual]

Destructor.

Definition at line 618 of file ChannelsSurprise.C.

### 10.181.3 Member Function Documentation

#### 10.181.3.1 [ColorBandChannelSurprise](#) & [MultiColorBandChannelSurprise](#)::band (const [uint](#) *idx*) const [virtual]

Wraps [ComplexChannel](#)'s [subChan\(\)](#) and casts to [ColorBandChannelSurprise](#).

Reimplemented from [MultiColorBandChannel](#).

Definition at line 623 of file [ChannelsSurprise.C](#).

References [ComplexChannel::subChan\(\)](#), and [uint](#).

#### 10.181.3.2 [void MultiColorBandChannelSurprise::buildSubChans \(\)](#) [protected, virtual]

(re-)build our subchannels

Reimplemented from [MultiColorBandChannel](#).

Definition at line 596 of file [ChannelsSurprise.C](#).

References [ComplexChannel::addSubChan\(\)](#), [ModelComponent::exportOptions\(\)](#), [ModelParam< uint >::getVal\(\)](#), [makeShared\(\)](#), [OPTEXP\\_NONE](#), [ModelComponent::removeAllSubComponents\(\)](#), [ChannelBase::setHomogeneous\(\)](#), and [uint](#).

Referenced by [MultiColorBandChannelSurprise\(\)](#).

The documentation for this class was generated from the following files:

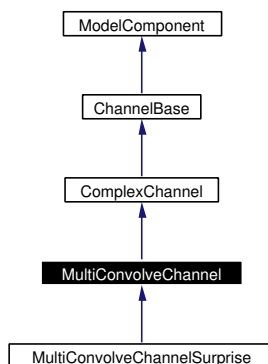
- [ChannelsSurprise.H](#)
- [ChannelsSurprise.C](#)

## 10.182 MultiConvolveChannel Class Reference

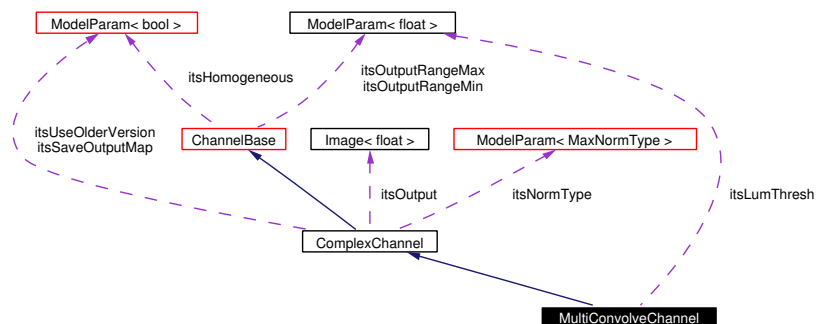
A generic multi-convolution channel.

```
#include <Channels.H>
```

Inheritance diagram for MultiConvolveChannel:



Collaboration diagram for MultiConvolveChannel:



### Public Member Functions

- `MultiConvolveChannel (ModelManager &mgr)`  
*Construct with standard params.*
- `virtual ~MultiConvolveChannel ()`  
*Destructor.*

## Protected Member Functions

- void [start1](#) ()  
*read filter definition and instantiate subchannels*
- virtual void [doInput](#) (const [Image](#)< [PixRGB](#)< float > > \*colimg, const [Image](#)< float > \*, const double t, const [Image](#)< byte > &clipMask)  
*MultiConvolveChannel requires only color input.*

## Protected Attributes

- [ModelParam](#)< std::string > [itsFilterFname](#)  
*file name for the filters*
- [ModelParam](#)< float > [itsLumThresh](#)  
*Luminance threshold (see [getRGBY\(\)](#)).*

### 10.182.1 Detailed Description

A generic multi-convolution channel.

This channel reads a filter definition file from disk (text format) and builds a number of [ConvolveChannel](#) subchannels. In the current implementation, only 4 subchannels are supported, and their inputs will be the red, green, blue and yellow color components of the image as given by [getRGBY\(\)](#). Each subchannel will convolve its input using the convolution filter definition given in the text file.

Definition at line 869 of file Channels.H.

### 10.182.2 Constructor & Destructor Documentation

#### 10.182.2.1 MultiConvolveChannel::MultiConvolveChannel ([ModelManager](#) & *mgr*)

Construct with standard params.

Definition at line 694 of file Channels.C.

References [ComplexChannel::addSubChan\(\)](#), and [CONVOLVE](#).

#### 10.182.2.2 MultiConvolveChannel::~MultiConvolveChannel () [virtual]

Destructor.

Definition at line 717 of file Channels.C.

### 10.182.3 Member Function Documentation

#### 10.182.3.1 void MultiConvolveChannel::doInput (const Image< PixRGB< float > > \* colimg, const Image< float > \*, const double t, const Image< byte > & clipMask) [protected, virtual]

MultiConvolveChannel requires only color input.

Implements [ChannelBase](#).

Definition at line 754 of file Channels.C.

References [ASSERT](#), [getRGBY\(\)](#), [ModelParam< float >::getVal\(\)](#), [ChannelBase::input\(\)](#), [itsLumThresh](#), [ComplexChannel::numChans\(\)](#), and [ComplexChannel::subChan\(\)](#).

#### 10.182.3.2 void MultiConvolveChannel::start1 () [protected, virtual]

read filter definition and instantiate subchannels

Reimplemented from [ModelComponent](#).

Definition at line 721 of file Channels.C.

References [itsFilterFname](#), [NO\\_INIT](#), [Image< T >::setVal\(\)](#), and [ComplexChannel::subChan\(\)](#).

### 10.182.4 Member Data Documentation

#### 10.182.4.1 ModelParam<std::string> MultiConvolveChannel::itsFilterFname [protected]

file name for the filters

Definition at line 879 of file Channels.H.

Referenced by [start1\(\)](#).



#### 10.182.4.2 [ModelParam<float>](#) [MultiConvolveChannel::itsLumThresh](#) [protected]

Luminance threshold (see [getRGBY\(\)](#)).

Definition at line 880 of file Channels.H.

Referenced by [doInput\(\)](#).

The documentation for this class was generated from the following files:

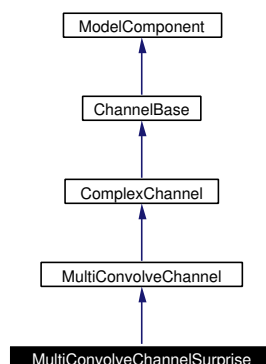
- [Channels.H](#)
- [Channels.C](#)

## 10.183 MultiConvolveChannelSurprise Class Reference

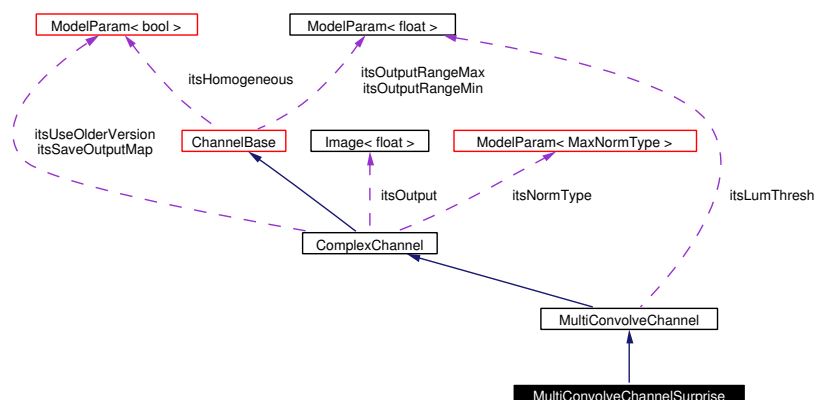
A generic multi-convolution channel.

```
#include <ChannelsSurprise.H>
```

Inheritance diagram for MultiConvolveChannelSurprise:



Collaboration diagram for MultiConvolveChannelSurprise:



### Public Member Functions

- [MultiConvolveChannelSurprise](#) ([ModelManager](#) &mgr)

*Construct with standard params.*

- virtual [~MultiConvolveChannelSurprise](#) ()

*Destructor.*

### 10.183.1 Detailed Description

A generic multi-convolution channel.

Definition at line 426 of file ChannelsSurprise.H.

### 10.183.2 Constructor & Destructor Documentation

#### 10.183.2.1 MultiConvolveChannelSurprise::MultiConvolveChannelSurprise ([ModelManager](#) & mgr)

Construct with standard params.

Definition at line 387 of file ChannelsSurprise.C.

References [ComplexChannel::addSubChan\(\)](#), [ModelComponent::removeAllSubComponents\(\)](#), and [ChannelBase::setHomogeneous\(\)](#).

#### 10.183.2.2 MultiConvolveChannelSurprise::~~MultiConvolveChannelSurprise () [virtual]

*Destructor.*

Definition at line 413 of file ChannelsSurprise.C.

The documentation for this class was generated from the following files:

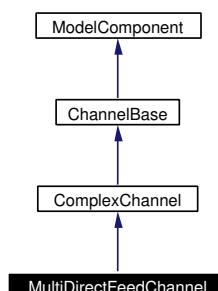
- [ChannelsSurprise.H](#)
- ChannelsSurprise.C

## 10.184 MultiDirectFeedChannel Class Reference

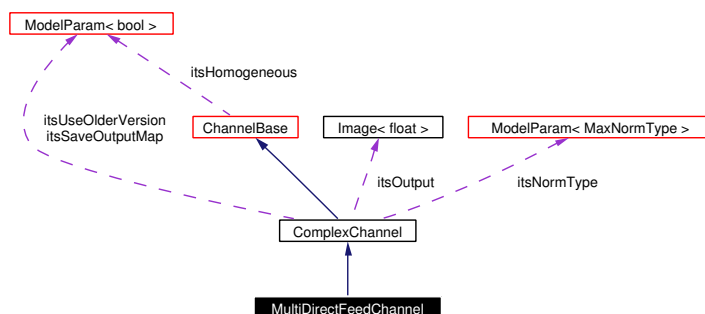
Multi Direct Feed Channel.

```
#include <DirectFeedChannel.H>
```

Inheritance diagram for MultiDirectFeedChannel:



Collaboration diagram for MultiDirectFeedChannel:



### Public Member Functions

- [MultiDirectFeedChannel](#) ([ModelManager](#) &mgr, [uint](#) num=1)  
*Constructor.*
- virtual [~MultiDirectFeedChannel](#) ()  
*Virtual destructor.*
- virtual void [setNumChans](#) ([uint](#) num)  
*reset the number of sub-channels*

- virtual void `inputPyramid` (`uint` ChanNum, const `ImageSet`< float > &pyramid, const double t)  
*Send an already computed pyramid as input to the sub-channel num.*
- virtual void `inputPyramidVector` (const `PyrVec` &pvec, const double t)  
*Send a vector of already computed pyramids as inputs to the channels.*
- virtual `DirectFeedChannel` & `directFeed` (`uint` num)  
*returns sub-channel number num*

## Protected Member Functions

- virtual `Image`< float > `combineOutputs` ()  
*Combines the outputs of subchannels.*
- virtual void `doInput` (const `Image`< `PixRGB`< float > > \*colimg, const `Image`< float > \*bwimg, const double t, const `Image`< byte > &clipMask)  
*must be called with either colimg or bwimg to obtain the dims*

### 10.184.1 Detailed Description

Multi Direct Feed Channel.

Definition at line 219 of file DirectFeedChannel.H.

### 10.184.2 Constructor & Destructor Documentation

#### 10.184.2.1 MultiDirectFeedChannel::MultiDirectFeedChannel (`ModelManager` & mgr, `uint` num = 1)

Constructor.

**Parameters:**

- mgr* our `ModelManager` (see `ModelManager.H`)
- numChans* number of sub-channels

Definition at line 367 of file DirectFeedChannel.C.

References MULTIDIRECTFEED, setNumChans(), and uint.

### 10.184.2.2 **MultiDirectFeedChannel::~~MultiDirectFeedChannel ()** [virtual]

Virtual destructor.

Definition at line 373 of file DirectFeedChannel.C.

## 10.184.3 Member Function Documentation

### 10.184.3.1 **Image< float > MultiDirectFeedChannel::combineOutputs ()** [protected, virtual]

Combines the outputs of subchannels.

Reimplemented from [ComplexChannel](#).

Definition at line 422 of file DirectFeedChannel.C.

References [ASSERT](#), [directFeed\(\)](#), [ComplexChannel::getMapDims\(\)](#), [DirectFeedChannel::getOutput\(\)](#), [ModelParam< MaxNormType >::getVal\(\)](#), [ModelParam< float >::getVal\(\)](#), [maxNormalize\(\)](#), [ComplexChannel::numChans\(\)](#), [uint](#), and [ZEROS](#).

### 10.184.3.2 **DirectFeedChannel & MultiDirectFeedChannel::directFeed (uint num)** [virtual]

returns sub-channel number num

Definition at line 414 of file DirectFeedChannel.C.

References [ASSERT](#), [ComplexChannel::numChans\(\)](#), [ComplexChannel::subChan\(\)](#), and [uint](#).

Referenced by [combineOutputs\(\)](#), [doInput\(\)](#), and [inputPyramid\(\)](#).

### 10.184.3.3 **void MultiDirectFeedChannel::doInput (const Image< PixRGB< float > > \* colimg, const Image< float > \* bwimg, const double t, const Image< byte > & clipMask)** [protected, virtual]

must be called with either colimg or bwimg to obtain the dims

After each call of input, inputPyramidVector or inputPyramid must be called.

Implements [ChannelBase](#).

Definition at line 440 of file DirectFeedChannel.C.

References [ASSERT](#), [directFeed\(\)](#), [ChannelBase::input\(\)](#), [ComplexChannel::numChans\(\)](#), and [uint](#).

**10.184.3.4 void MultiDirectFeedChannel::inputPyramid (uint ChanNum, const ImageSet< float > & pyramid, const double t) [virtual]**

Send an already computed pyramid as input to the sub-channel num.

Definition at line 396 of file DirectFeedChannel.C.

References ASSERT, directFeed(), DirectFeedChannel::inputPyramid(), ComplexChannel::numChans(), and uint.

Referenced by inputPyramidVector().

**10.184.3.5 void MultiDirectFeedChannel::inputPyramidVector (const PyrVec & pvec, const double t) [virtual]**

Send a vector of already computed pyramids as inputs to the channels.

Definition at line 405 of file DirectFeedChannel.C.

References ASSERT, inputPyramid(), ComplexChannel::numChans(), PyrVec, and uint.

**10.184.3.6 void MultiDirectFeedChannel::setNumChans (uint num) [virtual]**

reset the number of sub-channels

Definition at line 377 of file DirectFeedChannel.C.

References ComplexChannel::addSubChan(), makeShared(), ComplexChannel::numChans(), ModelComponent::removeSubComponent(), and uint.

Referenced by MultiDirectFeedChannel().

The documentation for this class was generated from the following files:

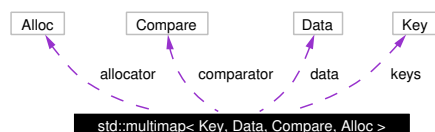
- [DirectFeedChannel.H](#)
- [DirectFeedChannel.C](#)

## 10.185 `std::multimap< Key, Data, Compare, Alloc >` Class Template Reference

The STL multimap container class. Use with `#include` .

`#include <dummySTL.H>`

Collaboration diagram for `std::multimap< Key, Data, Compare, Alloc >`:



### Public Attributes

- Key **keys**
- Data **data**
- Compare **comparator**
- Alloc **allocator**

#### 10.185.1 Detailed Description

`template<class Key, class Data, class Compare, class Alloc> class  
std::multimap< Key, Data, Compare, Alloc >`

The STL multimap container class. Use with `#include` .

See [SGI's documentation](#) of `std::multimap`.

Definition at line 56 of file `dummySTL.H`.

The documentation for this class was generated from the following file:

- [dummySTL.H](#)

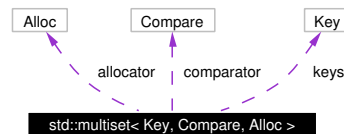


## 10.186 `std::multiset< Key, Compare, Alloc >` Class Template Reference

The STL multiset container class. Use with `#include` .

`#include <dummySTL.H>`

Collaboration diagram for `std::multiset< Key, Compare, Alloc >`:



### Public Attributes

- Key **keys**
- Compare **comparator**
- Alloc **allocator**

### 10.186.1 Detailed Description

`template<class Key, class Compare, class Alloc> class std::multiset< Key, Compare, Alloc >`

The STL multiset container class. Use with `#include` .

See [SGI's documentation](#) of `std::multiset`.

Definition at line 50 of file `dummySTL.H`.

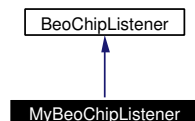
The documentation for this class was generated from the following file:

- [dummySTL.H](#)

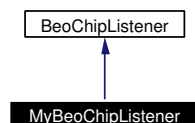
## 10.187 MyBeoChipListener Class Reference

Our own little [BeoChipListener](#).

Inheritance diagram for MyBeoChipListener:



Collaboration diagram for MyBeoChipListener:



### Public Member Functions

- virtual void [event](#) (const [BeoChipEventType](#) t, const int valint, const float valfloat)

*An event was received.*

### Public Attributes

- volatile int **rawpwm** [2]
- volatile int **rawadc** [2]
- volatile int **kbd**

### 10.187.1 Detailed Description

Our own little [BeoChipListener](#).

Definition at line 44 of file test-BeoChip.C.

## 10.187.2 Member Function Documentation

### 10.187.2.1 `virtual void MyBeoChipListener::event (const BeoChipEventType t, const int valint, const float valfloat) [inline, virtual]`

An event was received.

This function will be called by the [BeoChip](#) once you have registered your [BeoChipListener](#) derivative with the [BeoChip](#) using [BeoChip::setListener\(\)](#). Beware that the call to [event\(\)](#) will ve from a thread running in the [BeoChip](#) and thus will operate in parallel with your main thread and main loop. See [test-BeoChip.C](#) for an example of how to synchronize a main loop with these asynchronous event calls. The following data will be passed along with the event type:

| Type       | valint                     | valfloat                    |
|------------|----------------------------|-----------------------------|
| NONE       | 0                          | 0.0F                        |
| PWM0       | pulse 0 width, 1.6us units | calibrated width in [-1..1] |
| PWM1       | pulse 1 width, 1.6us units | calibrated width in [-1..1] |
| KBD        | 5 LSB show the 5 keys      | 0.0F                        |
| ADC0       | A/D 0 value                | calibrated value in [-1..1] |
| ADC1       | A/D 0 value                | calibrated value in [-1..1] |
| all others | 0                          | 0.0F                        |

Implements [BeoChipListener](#).

Definition at line 53 of file [test-BeoChip.C](#).

The documentation for this class was generated from the following file:

- [test-BeoChip.C](#)

## 10.188 NewClass Class Reference

Definition and access functions for a new class.

```
#include <NewClass.H>
```

### Public Member Functions

- [NewClass](#) (const int x, const int y, const char \*name)

*Constructor.*

- [~NewClass](#) ()

*Destructor.*

- void [doStuff](#) ()

*Do some NewClass stuff.*

### 10.188.1 Detailed Description

Definition and access functions for a new class.

Describe here what your class does and how it works. Please remember: no more than 80 chars/line; absolutely no code inside the class definition (put your inlined methods after and outside the class definition); and please see the README for other guidelines.

Definition at line 48 of file NewClass.H.

### 10.188.2 Constructor & Destructor Documentation

#### 10.188.2.1 NewClass::NewClass (const int x, const int y, const char \* name)

Constructor.

Build a new NewClass object with the following characteristics:

**Parameters:**

**x** horizontal coordinate

**y** vertical coordinate

**name** name for this instance of NewClass

### 10.188.2.2 NewClass::~~NewClass ()

Destructor.

## 10.188.3 Member Function Documentation

### 10.188.3.1 void NewClass::doStuff () [inline]

Do some NewClass stuff.

Write a longer description here

Definition at line 75 of file NewClass.H.

The documentation for this class was generated from the following file:

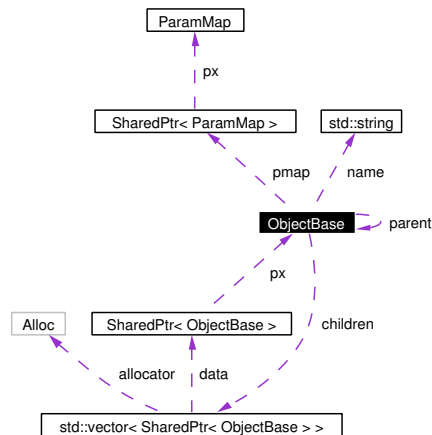
- [NewClass.H](#)

## 10.189 ObjectBase Class Reference

ObjectBase represents the various object modules for recognition.

```
#include <ObjectBase.H>
```

Collaboration diagram for ObjectBase:



### Name and VisualFeature functions

- const **std::string** **descriptiveName** () const
- void **setDescription** (const **std::string** name)  
Change this channel's descriptive name.
- const int **tagName** () const  
Get a programmer-usable tag name for this object.
- void **setTagName** (int tag)
- const **SharedPtr< ParamMap >** **getParam** () const
- void **setParam** (**SharedPtr< ParamMap >** pmap)
- const double **getMismatch** () const
- void **setMismatch** (double mismatch)
- const double **getLikelihood** () const
- void **setLikelihood** (double mismatch)
- const double **getOutside\_range** () const
- void **setOutside\_range** (double mismatch)
- const double **getScore** () const
- void **setScore** (double mismatch)

- `ObjectBase * getParent ()`
- `void setParent (ObjectBase *parent)`
- `const std::vector< SharedPtr< ParamMap > > getChildren () const`
- `void loadChildren (const char *filename)`
- `bool recognize (SharedPtr< VisualCortex > vcx, std::map< double, SharedPtr< ObjectBase >, std::greater< double > > &matches)`
- `bool compareAll (SharedPtr< VisualCortex > vcx, std::map< double, SharedPtr< ObjectBase >, std::greater< double > > &matches)`  
*compare vcx with all children recursively in a BFS*
- `void compareChildren (SharedPtr< VisualCortex > vcx, std::map< double, SharedPtr< ObjectBase >, std::greater< double > > &matches)`
- `SharedPtr< ObjectBase > bestMatch (std::map< double, SharedPtr< ObjectBase >, std::greater< double > > &matches) const`
- `void addToKB (SharedPtr< ObjectBase > best_match, SharedPtr< VisualCortex > vcx)`
- `void addChild (SharedPtr< ObjectBase > child)`
- `void print ()`
- `void getScaleInvariantPmap (SharedPtr< ParamMap > abs_pmap)`
- `void getRotationInvariantPmap (SharedPtr< ParamMap > abs_pmap)`

## Public Member Functions

- `ObjectBase (const std::string descrName, int tagName, SharedPtr< ParamMap > pmap)`  
*Default constructor.*
- `virtual ~ObjectBase ()`  
*Virtual destructor ensures proper destruction of derived classes.*

### 10.189.1 Detailed Description

ObjectBase represents the various object modules for recognition.

Definition at line 66 of file ObjectBase.H.

### 10.189.2 Constructor & Destructor Documentation

#### 10.189.2.1 ObjectBase::ObjectBase (const std::string descrName, int tagName, SharedPtr< ParamMap > pmap)

Default constructor.

Definition at line 41 of file ObjectBase.C.

#### 10.189.2.2 `ObjectBase::~~ObjectBase ()` [virtual]

Virtual destructor ensures proper destruction of derived classes.

Definition at line 58 of file ObjectBase.C.

### 10.189.3 Member Function Documentation

#### 10.189.3.1 `bool ObjectBase::compareAll (SharedPtr< VisualCortex > vcx, std::map< double, SharedPtr< ObjectBase >, std::greater< double > > & matches)`

compare vcx with all children recursively in a BFS

Definition at line 232 of file ObjectBase.C.

References uint.

#### 10.189.3.2 `void ObjectBase::setDescriptiveName (const std::string name)` [inline]

Change this channel's descriptive name.

Definition at line 183 of file ObjectBase.H.

#### 10.189.3.3 `const int ObjectBase::tagName () const` [inline]

Get a programmer-usable tag name for this object.

This is intended to be unique among channel objects, so that this string could be used for associate-array indexing, for example.

Definition at line 188 of file ObjectBase.H.

The documentation for this class was generated from the following files:

- ObjectBase.H
- ObjectBase.C



## 10.190 OldFrameSeries Class Reference

Manages a series of input or output frames for vision.C::main().

### Public Types

### Public Member Functions

- **OldFrameSeries** (const char \*name\_)
- void **scan** (const char \*arg)
- State **update** (double mstime, double msdelay)
- bool **isMultiframe** () const
- double **delay** () const
- int **frame** () const
- int **minframe** () const
- int **maxframe** () const
- bool **finished** () const

### 10.190.1 Detailed Description

Manages a series of input or output frames for vision.C::main().

It keeps track of a range of frames ([first..last]) and an interframe delay in milliseconds. It can grab this information from a string argument with its scan() function, and its key behavior is the update() function, which takes the current simulation time in milliseconds and the simulation step in ms, and returns a status code indicating whether the frame series (a) was unchanged, (b) will require that the next frame be loaded within the current simulation step, or (c) is finished.

Definition at line 363 of file vision.C.

The documentation for this class was generated from the following file:

- [vision.C](#)

## 10.191 Omni< T > Class Template Reference

omni directional correction tool

```
#include <Omni.H>
```

### Public Member Functions

- [Omni \(\)](#)  
*Omni correction object constructor.*
- [~Omni \(\)](#)  
*Omni correction object destructor.*
- [Image< T > run](#) (const [Image< T >](#) &image, [readConfig](#) &config)  
*Run omni correction on an [Image](#), return a new [Image](#).*

### 10.191.1 Detailed Description

```
template<class T> class Omni< T >
```

omni directional correction tool

Definition at line 51 of file Omni.H.

### 10.191.2 Constructor & Destructor Documentation

#### 10.191.2.1 template<class T> [Omni< T >::Omni \(\)](#)

Omni correction object constructor.

Definition at line 95 of file Omni.C.

#### 10.191.2.2 template<class T> [Omni< T >::~~Omni \(\)](#)

Omni correction object destructor.

Definition at line 100 of file Omni.C.

### 10.191.3 Member Function Documentation

#### 10.191.3.1 `template<class T> Image< T > Omni< T >::run (const Image< T > & image, readConfig & config)`

Run omni correction on an [Image](#), return a new [Image](#).

This method will call both Omni corrections for general and special correction on an [Image](#). The two methods can be found in [Image.H](#) as `omniCorrectGen` and `omniCorrectSP`. If a pinhole image is used `omniCorrectSP` may be skipped.

**Parameters:**

*image* This is the image to be corrected of type [Image](#)

*config* This is the config file for processing of type [readConfig](#)

Definition at line 106 of file `Omni.C`.

References `readConfig::getItemValueF()`, `Image< PixRGB< float > >::getWidth()`, `image`, `omniCorrectGen()`, `omniDenebulize()`, and `Raster::Visu()`.

The documentation for this class was generated from the following files:

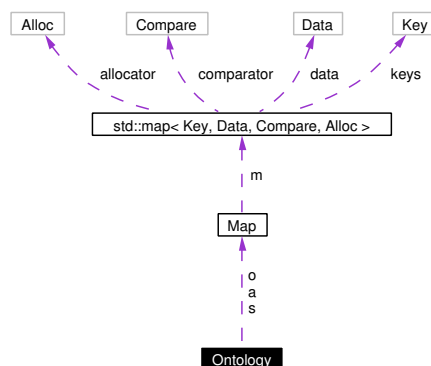
- [Omni.H](#)
- [Omni.C](#)

## 10.192 Ontology Class Reference

Definition and access functions for Ontology.

```
#include <Agent.H>
```

Collaboration diagram for Ontology:



### Public Member Functions

- [Ontology](#) ()  
*Constructor.*
- [~Ontology](#) ()  
*Destructor.*
- void [insert\\_element](#) (char c, [std::string](#) name, [Entity](#) &entity)  
*insert an element into the ontology*
- const [Entity](#) \* [get](#) (char c, [std::string](#) name)  
*find the element in ontology; returns pointer to retrieved entity*
- void [delete\\_element](#) (char c, [std::string](#) name)  
*delete an element from the ontology*

### 10.192.1 Detailed Description

Definition and access functions for Ontology.

Ontology class consists of the object, subject and action ontology.

Definition at line 67 of file Agent.H.

## 10.192.2 Constructor & Destructor Documentation

### 10.192.2.1 Ontology::Ontology () [inline]

Constructor.

Ensure that your working directory has XML files called "object\_file.xml", "subject\_file.xml" and "action\_file.xml".

Definition at line 114 of file Agent.H.

References Map::create\_Map().

### 10.192.2.2 Ontology::~~Ontology () [inline]

Destructor.

Definition at line 110 of file Agent.H.

## 10.192.3 Member Function Documentation

### 10.192.3.1 void Ontology::delete\_element (char *c*, std::string *name*) [inline]

delete an element from the ontology

#### Parameters:

- c* enter 'o' or 's' or 'a' depending on ontology
- name* enter the name of entity to be deleted

Definition at line 159 of file Agent.H.

References Map::delete\_element().

### 10.192.3.2 const Entity \* Ontology::get (char *c*, std::string *name*) [inline]

find the element in ontology; returns pointer to retrieved entity

#### Parameters:

- c* enter 'o' or 's' or 'a' depending on ontology
- name* enter the name of entity to be retrieved

Definition at line 145 of file Agent.H.

References Map::get().

#### 10.192.3.3 **void Ontology::insert\_element** (char *c*, **std::string** *name*, **Entity** & *entity*) [inline]

insert an element into the ontology

##### **Parameters:**

*c* enter 'o' or 's' or 'a' depending on ontology

*name* enter the name of entity to be inserted

*entity* enter a reference to entity to be inserted

Definition at line 126 of file Agent.H.

References Map::insert\_element().

The documentation for this class was generated from the following file:

- Agent.H

## 10.193 OptionParser Class Reference

Manages the parsing of argc/argv command-line arguments.

```
#include <OptionParser.H>
```

### Public Member Functions

- [OptionParser](#) (const char \*appName, int argc, const char \*\*argv, const [OptionSet](#) &optset)  
*Construct with the argc/argv from main(), and an [OptionSet](#).*
- [~OptionParser](#) ()  
*Destructor.*
- void [destroy](#) ()  
*Destroy our representation and free any memory.*
- void [addAlias](#) (const char \*longName, char shortName, const char \*aliasArgs, const char \*description=0, const char \*argDescript=0)  
*Add an alias: a single option can alias a sequence of multiple args.*
- void [setOtherHelp](#) (const char \*usage)  
*Provide a message describing additional arguments unknown to [OptionSet](#).*
- void [printUsage](#) ()  
*Print an abbreviated usage message to stderr.*
- void [printHelp](#) ()  
*Print a detailed usage message to stderr.*
- void [printError](#) (int errorCode)  
*Print an error message to stderr describing errorCode.*
- int [getNextOpt](#) ()  
*Get the code indicating the next option.*
- const char \* [getCurrentOptArg](#) ()  
*Returns the argument, if any, to the most recently parsed option.*
- const char \*\* [getLeftoverArgs](#) () const  
*Get an argv-style null-terminated array of the remaining args.*

- `const char * getOneLeftoverArg () const`

*Get the next leftover arg, or return null if no args are left.*

### 10.193.1 Detailed Description

Manages the parsing of argc/argv command-line arguments.

Works in collaboration with [OptionSet](#). This is a thin-ish wrapper around the *popt* library.

Definition at line 165 of file `OptionParser.H`.

### 10.193.2 Constructor & Destructor Documentation

#### 10.193.2.1 `OptionParser::OptionParser (const char * appName, int argc, const char ** argv, const OptionSet & optset)`

Construct with the argc/argv from `main()`, and an [OptionSet](#).

The [OptionSet](#) must be fully configured already when it is passed to the `OptionParser` constructor; any changes to the [OptionSet](#) *after* being passed to the constructor will **NOT** be reflected in the `OptionParser`.

Definition at line 170 of file `OptionParser.C`.

#### 10.193.2.2 `OptionParser::~~OptionParser ()`

Destructor.

Definition at line 183 of file `OptionParser.C`.

References `destroy()`.

### 10.193.3 Member Function Documentation

#### 10.193.3.1 `void OptionParser::addAlias (const char * longName, char shortName, const char * aliasArgs, const char * description = 0, const char * argDescript = 0)`

Add an alias: a single option can alias a sequence of multiple args.

##### Parameters:

***longName*** The long-form name for the alias (can be NULL).

***shortName*** The short-form name for the alias (can be ”).



***aliasArgs*** A string containing one or more arguments for which the alias can stand in.

***description*** A description of what the alias option does (can be NULL).

***argDescrip*** A description of the alias option's argument, if any (can be NULL).

For example, `parser.addAlias("--my-settings", '-m', "--size 64x64 --delay 30");`

Definition at line 197 of file `OptionParser.C`.

References `ASSERT`.

Referenced by `main()`.

#### 10.193.3.2 void OptionParser::destroy ()

Destroy our representation and free any memory.

It is NOT necessary to call this manually, since it will be taken care of by the destructor, but this can ensure that memory is freed once options are finished being parsed and before any serious work starts to get done in `main()`.

Definition at line 189 of file `OptionParser.C`.

References `ASSERT`.

Referenced by `main()`, and `~OptionParser()`.

#### 10.193.3.3 const char \* OptionParser::getCurrentOptArg ()

Returns the argument, if any, to the most recently parsed option.

Will return NULL if no argument was specified to the most recently parsed option.

Definition at line 277 of file `OptionParser.C`.

Referenced by `main()`.

#### 10.193.3.4 const char \*\* OptionParser::getLeftoverArgs () const

Get an argv-style null-terminated array of the remaining args.

Definition at line 283 of file `OptionParser.C`.

#### 10.193.3.5 int OptionParser::getNextOpt ()

Get the code indicating the next option.

Typically, you will want to call `getNextOpt()` repeatedly in a loop until it returns a negative value. A return value of -1 means that parsing succeeded, and all the options

have been processed. A return value of -2 or less means that an error occurred; you can then call [printError\(\)](#) with that error code to display a message describing the error. Note that some options are handled invisibly (i.e. without returning from [getNextOpt\(\)](#)); this can happen if the option is configured to just trigger an automatic option.

Definition at line 271 of file OptionParser.C.

Referenced by main().

#### **10.193.3.6    `const char * OptionParser::getOneLeftoverArg () const`**

Get the next leftover arg, or return null if no args are left.

Definition at line 289 of file OptionParser.C.

Referenced by main().

#### **10.193.3.7    `void OptionParser::printError (int errorCode)`**

Print an error message to stderr describing *errorCode*.

Definition at line 262 of file OptionParser.C.

Referenced by main().

#### **10.193.3.8    `void OptionParser::printHelp ()`**

Print a detailed usage message to stderr.

Definition at line 256 of file OptionParser.C.

#### **10.193.3.9    `void OptionParser::printUsage ()`**

Print an abbreviated usage message to stderr.

Definition at line 250 of file OptionParser.C.

Referenced by main().

#### **10.193.3.10    `void OptionParser::setOtherHelp (const char * usage)`**

Provide a message describing additional arguments unknown to [OptionSet](#).

These would typically be arguments that are not associated with any particular option.

Definition at line 244 of file OptionParser.C.

Referenced by main().

The documentation for this class was generated from the following files:

- [OptionParser.H](#)
- [OptionParser.C](#)

## 10.194 OptionSet Class Reference

Represents a set of available command-line options.

```
#include <OptionParser.H>
```

### Public Member Functions

- [OptionSet](#) (bool withAutohelp=true)  
*Construct, optionally with an automatic "--help" option.*
- [~OptionSet](#) ()  
*Destructor.*
- void [add](#) (const popOption &opt)  
*Add a popOption to the internal array.*
- void [add](#) (const char \*longName, char shortName, int argInfo=0, void \*argVariable=0, int val=0, const char \*description=0, const char \*argDescript=0)  
*Add an option description to the internal array.*
- void [addAutohelp](#) ()  
*Include an automatic "--help" option in the option set.*
- const popOption \* [opts](#) () const  
*Get the internal popOption array, e.g. for building an OptionsParser.*
- void [debugDump](#) () const  
*For testing/debugging only.*

### 10.194.1 Detailed Description

Represents a set of available command-line options.

Collaborates with [OptionParser](#) to facilitate semi-automated command-line argument processing. This is a thin-ish wrapper around the *popt* library. Basically it provides a dynamic way to build up the popOption array that is needed to build a popContext, or an OptionsParser, which is our wrapper around popContext.

Definition at line 51 of file OptionParser.H.

## 10.194.2 Constructor & Destructor Documentation

### 10.194.2.1 OptionSet::OptionSet (bool *withAutohelp* = true) [explicit]

Construct, optionally with an automatic "--help" option.

Definition at line 64 of file OptionParser.C.

References `addAutohelp()`.

### 10.194.2.2 OptionSet::~OptionSet ()

Destructor.

Definition at line 75 of file OptionParser.C.

## 10.194.3 Member Function Documentation

### 10.194.3.1 void OptionSet::add (const char \* *longName*, char *shortName*, int *argInfo* = 0, void \* *argVariable* = 0, int *val* = 0, const char \* *description* = 0, const char \* *argDescript* = 0)

Add an option description to the internal array.

#### Parameters:

***longName*** The long-form option string, can be NULL. You should **NOT** explicitly include the double-dash in this string; the double-dash will automatically be prepended by the underlying *popt* library.

***shortName*** The short-form option char, can be ". At least one of the short- and long-form options must be non-empty. This char is just the single character that specifies the option; no leading dash is necessary (the dash is automatically prepended by the underlying *popt* library.

***argInfo*** What kind of arg is expected after the option if the option is seen on command-line. This is given as a series of bitwise-or'ed flags:

| Value                  | Description                         | arg Type (cast to void*) |
|------------------------|-------------------------------------|--------------------------|
| <b>POPT_ARG_NONE</b>   | No argument expected                | int*                     |
| <b>POPT_ARG_STRING</b> | No type checking to be performed    | char**                   |
| <b>POPT_ARG_INT</b>    | An integer argument is expected     | int*                     |
| <b>POPT_ARG_LONG</b>   | A long integer is expected          | long*                    |
| <b>POPT_ARG_FLOAT</b>  | An float argument is expected       | float*                   |
| <b>POPT_ARG_DOUBLE</b> | A double argument is expected       | double*                  |
| <b>POPT_ARG_VAL</b>    | Integer value taken from <i>val</i> | int*                     |

In the case of **POPT\_ARG\_NONE**, the variable pointed to by *arg* is set to 1 when the option is seen on the command-line.

In the case of **POPT\_ARG\_STRING**, **POPT\_ARG\_INT**, **POPT\_ARG\_LONG**, **POPT\_ARG\_FLOAT**, and **POPT\_ARG\_DOUBLE**, the argument from the command line will be converted to the expected type, and the variable pointed to by *arg* will be set to this value.

In the case of **POPT\_ARG\_VAL**, the variable pointed to by *arg* will be set to the value given in *val*. Or, one of the following additional flags may be optionally paired with **POPT\_ARG\_VAL** (in this case, no argument is expected on the command-line, but the variable pointed to by *argVariable* will be updated in some way with the value in *val*): **POPT\_ARGFLAG\_OR**, **POPT\_ARGFLAG\_AND**, **POPT\_ARGFLAG\_XOR**, **POPT\_BIT\_SET**, **POPT\_BIT\_CLR**, all of which perform the predictable bitwise operations (equivalent to *\*argVariable op= val*).

#### Parameters:

***argVariable*** Pointer to variable to be updated with option argument. If NULL, then no special action is taken when the option is seen on the command-line.

***val*** Behaves differently depending on *argInfo*.

- If *argInfo* does not include **POPT\_ARG\_VAL**, then *val* specifies the value that should be returned from [OptionParser::getNextOpt\(\)](#) when the option is seen on the command-line, unless *val* is 0, in which case [OptionParser::getNextOpt\(\)](#) does not return a value for this option, instead moving directly to the next command-line argument.
- If *argInfo* does include **POPT\_ARG\_VAL**, then *val* specifies the value with which to update *argVariable*. In this latter case, nothing will be returned from [OptionParser::getNextOpt\(\)](#) for this option, since it is processed fully automatically.

***description*** A description of what the option does, can be NULL.

***argDescrip*** A description of the option's argument, if any (can be NULL).

The parameters to this function are the same as the fields of the `poptOption` struct; see *popt*'s documentation (man pages) for more detailed info.

Definition at line 109 of file `OptionParser.C`.

References `add()`, and `ASSERT`.

#### 10.194.3.2 void OptionSet::add (const popOption & opt)

Add a `poptOption` to the internal array.

Definition at line 88 of file `OptionParser.C`.

Referenced by `add()`, `addAutohelp()`, and `main()`.

#### 10.194.3.3 void OptionSet::addAutohelp ()

Include an automatic "--help" option in the option set.

This automatically configures things so that if "--help" is seen on the command-line, a list of available options will be printed and the program will exit.

Definition at line 126 of file `OptionParser.C`.

References `add()`.

Referenced by `OptionSet()`.

#### 10.194.3.4 void OptionSet::debugDump () const

For testing/debugging only.

Definition at line 148 of file `OptionParser.C`.

References `uint`.

#### 10.194.3.5 const popOption \* OptionSet::opts () const

Get the internal `poptOption` array, e.g. for building an `OptionsParser`.

Definition at line 142 of file `OptionParser.C`.

The documentation for this class was generated from the following files:

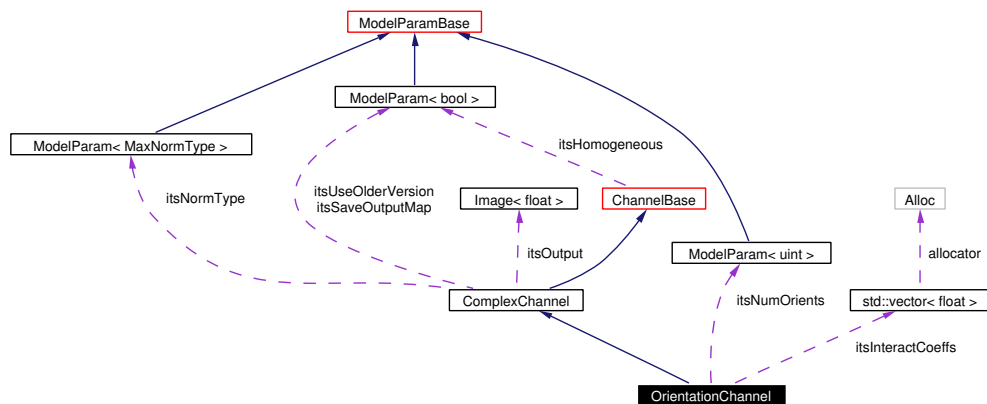
- [OptionParser.H](#)
- [OptionParser.C](#)

## 10.195 OrientationChannel Class Reference

A composite channel containing a set of orientation-tuned gabor channels.

```
#include <Channels.H>
```

Inheritance diagram for OrientationChannel: Collaboration diagram for OrientationChannel:



### Public Types

- enum [InteractType](#) { NONE = 0, SUB\_MEAN = 1, CUSTOM = 2 }  
*the possible interaction types between orientations*

### Public Member Functions

- [OrientationChannel](#) ([ModelManager](#) &mgr)  
*Construct with standard params.*
- virtual [~OrientationChannel](#) ()  
*Destructor.*
- virtual void [exportOptions](#) (const int whatToExport, const bool recurse=true)  
*Export some of our [ModelParam](#) parameters as command-line-options.*
- virtual [GaborChannel](#) & [gabor](#) (const [uint](#) idx) const  
*Wraps [ComplexChannel](#)'s [subChan\(\)](#) and casts the result to [GaborChannel](#).*



- virtual bool [setModelParamString](#) (const [std::string](#) &name, const [std::string](#) &value, const bool recurse=false, const bool warn=true)  
*Overload so that we can reconfigure when our params get changed.*
- [InteractType](#) [getInteractType](#) ()  
*returns the currently set interaction type between the orientation*
- void [setInteractType](#) ([InteractType](#) type)  
*sets the interaction type between the orientations*
- [std::vector](#)< float > [getInteractCoeffs](#) ()  
*returns the coefficients vector for interactions between orientations*
- void [setInteractCoeffs](#) ([std::vector](#)< float > &coeffs)  
*sets the coefficients vector for the interactions between orientations*
- virtual [Image](#)< float > [getOutput](#) ()  
*compute interactions, then call [ComplexChannel::getOutput\(\)](#)*

## Protected Member Functions

- void [doInteractions](#) ()  
*trigger the interaction computation between the gabor filters*
- virtual void [buildSubChans](#) ()  
*(re-)build our subchannels*
- virtual void [parseInteractString](#) (const [std::string](#) &value)  
*parse the string for the interaction type between orientations*
- virtual void [doInput](#) (const [Image](#)< [PixRGB](#)< float > > \*, const [Image](#)< float > \*bwimg, const double t, const [Image](#)< byte > &clipMask)  
*OrientationChannel requires only luminance input.*

## Protected Attributes

- [ModelParam](#)< uint > [itsNumOrients](#)  
*number of Gabor subchannels*
- int [itsExportOpts](#)

*keep track of what options to export in case we rebuild our subchans*

- [ModelParam](#)< [std::string](#) > [itsInteractString](#)

*type of interaction mode between orientations*

### 10.195.1 Detailed Description

A composite channel containing a set of orientation-tuned gabor channels.

This is an example of a homogeneous complex channel. Thus, the [JetSpec](#) representation will have an index for the subchannel number, and further indices for the various [Jet](#) components (which are identical for all subchannels).

Definition at line 1027 of file Channels.H.

### 10.195.2 Member Enumeration Documentation

#### 10.195.2.1 enum [OrientationChannel::InteractType](#)

the possible interaction types between orientations

Definition at line 1051 of file Channels.H.

Referenced by [getInteractType\(\)](#).

### 10.195.3 Constructor & Destructor Documentation

#### 10.195.3.1 [OrientationChannel::OrientationChannel](#) ([ModelManager](#) & *mgr*)

Construct with standard params.

Definition at line 1014 of file Channels.C.

References [buildSubChans\(\)](#), [OPTEXP\\_NONE](#), and [ORI](#).

#### 10.195.3.2 [OrientationChannel::~~OrientationChannel](#) () [virtual]

Destructor.

Definition at line 1137 of file Channels.C.

## 10.195.4 Member Function Documentation

### 10.195.4.1 void OrientationChannel::buildSubChans () [protected, virtual]

(re-)build our subchannels

Reimplemented in [OrientationChannelBeo](#), and [OrientationChannelSurprise](#).

Definition at line 1028 of file Channels.C.

References [ComplexChannel::addSubChan\(\)](#), [exportOptions\(\)](#), [ModelParam< uint >::getVal\(\)](#), [itsExportOpts](#), [itsNumOrients](#), [makeShared\(\)](#), [OPTEXP\\_NONE](#), [ModelComponent::removeAllSubComponents\(\)](#), [ChannelBase::setHomogeneous\(\)](#), and [uint](#).

Referenced by [OrientationChannel\(\)](#), and [setModelParamString\(\)](#).

### 10.195.4.2 void OrientationChannel::doInput (const [Image< PixRGB< float > > \\*](#), const [Image< float > \\*](#) *bwimg*, const double *t*, const [Image< byte > & clipMask](#)) [protected, virtual]

[OrientationChannel](#) requires only luminance input.

Implements [ChannelBase](#).

Definition at line 1167 of file Channels.C.

References [ASSERT](#), [gabor\(\)](#), [ChannelBase::input\(\)](#), [ComplexChannel::numChans\(\)](#), and [uint](#).

### 10.195.4.3 void OrientationChannel::doInteractions () [protected]

trigger the interaction computation between the gabor filters

Definition at line 1182 of file Channels.C.

References [ASSERT](#), [clampedDiff\(\)](#), [doLowThresh\(\)](#), [gabor\(\)](#), [SingleChannel::getOutput\(\)](#), [itsInteractString](#), [ComplexChannel::numChans\(\)](#), and [uint](#).

Referenced by [getOutput\(\)](#).

### 10.195.4.4 void OrientationChannel::exportOptions (const int *whatToExport*, const bool *recurse* = true) [virtual]

Export some of our [ModelParam](#) parameters as command-line-options.

See the base function in [ModelComponent.H](#) for info.

Reimplemented from [ComplexChannel](#).

Definition at line 1141 of file Channels.C.

References `ComplexChannel::exportOptions()`, `itsExportOpts`, `itsInteractString`, `itsNumOrients`, `OPTEXP_CORE`, and `ModelManager::requestOption()`.

Referenced by `OrientationChannelSurprise::buildSubChans()`, `OrientationChannel-Beo::buildSubChans()`, and `buildSubChans()`.

#### 10.195.4.5 **GaborChannel** & **OrientationChannel::gabor (const uint idx) const** [virtual]

Wraps `ComplexChannel's subChan()` and casts the result to **GaborChannel**.

Reimplemented in **OrientationChannelSurprise**.

Definition at line 1159 of file Channels.C.

References `ComplexChannel::subChan()`, and `uint`.

Referenced by `doInput()`, and `doInteractions()`.

#### 10.195.4.6 **std::vector< float > OrientationChannel::getInteractCoeffs ()**

returns the coefficients vector for interactions between orientations

Definition at line 1293 of file Channels.C.

#### 10.195.4.7 **OrientationChannel::InteractType OrientationChannel::getInteractType ()**

returns the currently set interaction type between the orientation

Definition at line 1285 of file Channels.C.

References `InteractType`.

#### 10.195.4.8 **Image< float > OrientationChannel::getOutput ()** [virtual]

compute interactions, then call **ComplexChannel::getOutput()**

Reimplemented from **ComplexChannel**.

Definition at line 1274 of file Channels.C.

References `doInteractions()`, and `ComplexChannel::getOutput()`.

**10.195.4.9 void OrientationChannel::parseInteractString (const [std::string](#) & *value*)** [protected, virtual]

parse the string for the interaction type between orientations

Definition at line 1053 of file Channels.C.

Referenced by setModelParamString().

**10.195.4.10 void OrientationChannel::setInteractCoeffs ([std::vector](#)< float > & *coeffs*)**

sets the coefficients vector for the interactions between orientations

Definition at line 1297 of file Channels.C.

**10.195.4.11 void OrientationChannel::setInteractType ([InteractType](#) *type*)**

sets the interaction type between the orientations

Definition at line 1289 of file Channels.C.

**10.195.4.12 bool OrientationChannel::setModelParamString (const [std::string](#) & *name*, const [std::string](#) & *value*, const bool *recurse* = false, const bool *warn* = true)** [virtual]

Overload so that we can reconfigure when our params get changed.

Reimplemented from [ModelComponent](#).

Definition at line 1115 of file Channels.C.

References [buildSubChans\(\)](#), [ModelParamBase::getName\(\)](#), [ModelParam< uint >::getVal\(\)](#), [itsInteractString](#), [itsNumOrients](#), [ComplexChannel::numChans\(\)](#), [parseInteractString\(\)](#), and [ModelComponent::setModelParamString\(\)](#).

**10.195.5 Member Data Documentation****10.195.5.1 int [OrientationChannel::itsExportOpts](#)** [protected]

keep track of what options to export in case we rebuild our subchans

Definition at line 1086 of file Channels.H.

Referenced by [buildSubChans\(\)](#), and [exportOptions\(\)](#).

**10.195.5.2** [ModelParam<std::string>](#) [OrientationChannel::itsInteractString](#)  
[protected]

type of interaction mode between orientations

Definition at line 1089 of file Channels.H.

Referenced by [doInteractions\(\)](#), [exportOptions\(\)](#), and [setModelParamString\(\)](#).

**10.195.5.3** [ModelParam<uint>](#) [OrientationChannel::itsNumOrients](#)  
[protected]

number of Gabor subchannels

Definition at line 1072 of file Channels.H.

Referenced by [buildSubChans\(\)](#), [exportOptions\(\)](#), and [setModelParamString\(\)](#).

The documentation for this class was generated from the following files:

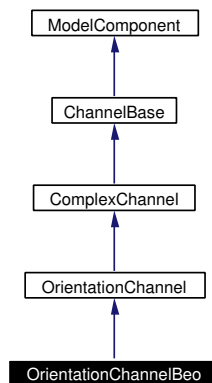
- [Channels.H](#)
- [Channels.C](#)

## 10.196 OrientationChannelBeo Class Reference

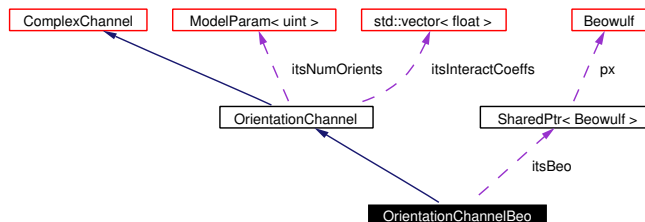
Beowulf version of [OrientationChannel](#).

```
#include <ChannelsBeo.H>
```

Inheritance diagram for OrientationChannelBeo:



Collaboration diagram for OrientationChannelBeo:



### Public Member Functions

- **OrientationChannelBeo** ([ModelManager](#) &mgr, const [SharedPtr< Beowulf >](#) &beow)

### Protected Member Functions

- virtual void [buildSubChans](#) ()  
(re-)build our subchannels

### 10.196.1 Detailed Description

[Beowulf](#) version of [OrientationChannel](#).

Definition at line 176 of file ChannelsBeo.H.

### 10.196.2 Member Function Documentation

#### 10.196.2.1 `void OrientationChannelBeo::buildSubChans ()` [protected, virtual]

(re-)build our subchannels

Reimplemented from [OrientationChannel](#).

Definition at line 266 of file ChannelsBeo.C.

References [ComplexChannel::addSubChan\(\)](#), [OrientationChannel::exportOptions\(\)](#), [ModelParam< uint >::getVal\(\)](#), [OPTEXP\\_NONE](#), [ModelComponent::removeAllSubComponents\(\)](#), [ChannelBase::setHomogeneous\(\)](#), and [uint](#).

The documentation for this class was generated from the following files:

- [ChannelsBeo.H](#)
- [ChannelsBeo.C](#)

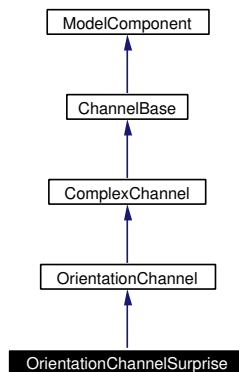


## 10.197 OrientationChannelSurprise Class Reference

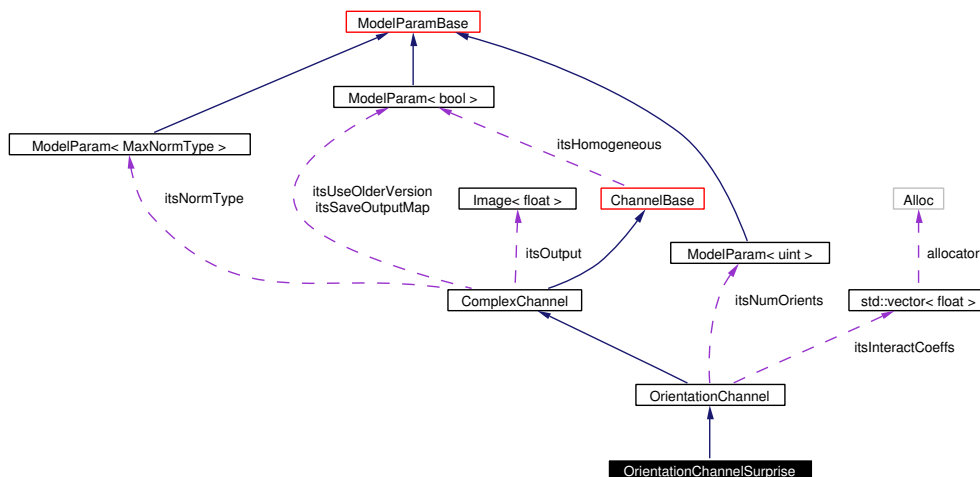
A composite channel containing a set of orientation-tuned gabor channels.

```
#include <ChannelsSurprise.H>
```

Inheritance diagram for OrientationChannelSurprise:



Collaboration diagram for OrientationChannelSurprise:



### Public Member Functions

- [OrientationChannelSurprise \(ModelManager &mgr\)](#)

*Construct with standard params.*

- virtual [~OrientationChannelSurprise](#) ()

*Destructor.*

- virtual [GaborChannelSurprise](#) & [gabor](#) (const [uint](#) idx) const

*Wraps ComplexChannel's [subChan\(\)](#) and casts to [GaborChannelSurprise](#).*

## Protected Member Functions

- virtual void [buildSubChans](#) ()

*(re-)build our subchannels*

### 10.197.1 Detailed Description

A composite channel containing a set of orientation-tuned gabor channels.

Definition at line 506 of file ChannelsSurprise.H.

### 10.197.2 Constructor & Destructor Documentation

#### 10.197.2.1 [OrientationChannelSurprise::OrientationChannelSurprise](#) ([ModelManager](#) & *mgr*)

Construct with standard params.

Definition at line 542 of file ChannelsSurprise.C.

References [buildSubChans\(\)](#).

#### 10.197.2.2 [OrientationChannelSurprise::~~OrientationChannelSurprise](#) () [virtual]

Destructor.

Definition at line 574 of file ChannelsSurprise.C.

### 10.197.3 Member Function Documentation

#### 10.197.3.1 `void OrientationChannelSurprise::buildSubChans ()` [protected, virtual]

(re-)build our subchannels

Reimplemented from [OrientationChannel](#).

Definition at line 551 of file ChannelsSurprise.C.

References [ComplexChannel::addSubChan\(\)](#), [OrientationChannel::exportOptions\(\)](#), [ModelParam< uint >::getVal\(\)](#), [makeShared\(\)](#), [OPTEXP\\_NONE](#), [ModelComponent::removeAllSubComponents\(\)](#), [ChannelBase::setHomogeneous\(\)](#), and [uint](#).

Referenced by [OrientationChannelSurprise\(\)](#).

#### 10.197.3.2 [GaborChannelSurprise](#) & `OrientationChannelSurprise::gabor` (const [uint](#) *idx*) const [virtual]

Wraps [ComplexChannel](#)'s [subChan\(\)](#) and casts to [GaborChannelSurprise](#).

Reimplemented from [OrientationChannel](#).

Definition at line 578 of file ChannelsSurprise.C.

References [ComplexChannel::subChan\(\)](#), and [uint](#).

The documentation for this class was generated from the following files:

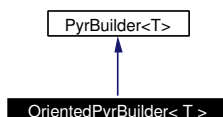
- [ChannelsSurprise.H](#)
- [ChannelsSurprise.C](#)

## 10.198 **OrientedPyrBuilder< T > Class Template Reference**

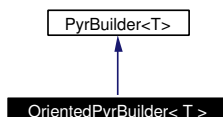
Builds pyramid based on oriented Laplacian filters.

```
#include <PyrBuilder.H>
```

Inheritance diagram for OrientedPyrBuilder< T >:



Collaboration diagram for OrientedPyrBuilder< T >:



### Public Member Functions

- [OrientedPyrBuilder](#) (const int filter\_size, const float theta=0.0F, const float intens=10.0F)  
*Construct with a given gabor filter orientation and intensity.*
- virtual [OrientedPyrBuilder< T > \\* clone \(\)](#) const  
*Cloning constructor:.*
- virtual [ImageSet< T > build](#) (const [Image< T >](#) &img, const int depth)  
*Builds an oriented pyramid based on img.*

### 10.198.1 Detailed Description

```
template<class T> class OrientedPyrBuilder< T >
```

Builds pyramid based on oriented Laplacian filters.

This is a fast approximation to Gabor convolution.

Definition at line 190 of file PyrBuilder.H.

## 10.198.2 Constructor & Destructor Documentation

**10.198.2.1** `template<class T> OrientedPyrBuilder< T >::OrientedPyrBuilder  
(const int filter_size, const float theta = 0.0F, const float intens =  
10.0F)`

Construct with a given gabor filter orientation and intensity.

Definition at line 177 of file PyrBuilder.C.

## 10.198.3 Member Function Documentation

**10.198.3.1** `template<class T> ImageSet< T > OrientedPyrBuilder< T  
>::build (const Image< T > & img, const int depth) [virtual]`

Builds an oriented pyramid based on *img*.

Implements **PyrBuilder**< T >.

Definition at line 185 of file PyrBuilder.C.

References `buildPyrOriented()`.

**10.198.3.2** `template<class T> OrientedPyrBuilder< T > *  
OrientedPyrBuilder< T >::clone () const [virtual]`

Cloning constructor:.

Implements **PyrBuilder**< T >.

Definition at line 189 of file PyrBuilder.C.

The documentation for this class was generated from the following files:

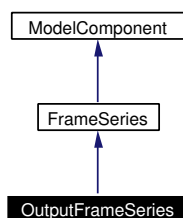
- [PyrBuilder.H](#)
- [PyrBuilder.C](#)

## 10.199 OutputFrameSeries Class Reference

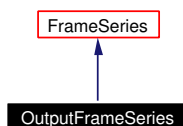
A [FrameSeries](#) with additional facilities for writing output frames.

```
#include <FrameSeries.H>
```

Inheritance diagram for OutputFrameSeries:



Collaboration diagram for OutputFrameSeries:



### Public Member Functions

- [OutputFrameSeries](#) ([ModelManager](#) &mgr, const [std::string](#) &descr-Name="Frame Series", const [std::string](#) &tagName="FrameSeries")

*Constructor.*

- virtual [~OutputFrameSeries](#) ()

*Destructor.*

- void [writeRGB](#) (const [Image](#)< [PixRGB](#)< [byte](#) > > &image, const [RasterFile-Format](#) ft, const int framenum, const char \*otherstem,...)

*Write a frame to disk.*

- void [writeRGB](#) (const [Image](#)< [PixRGB](#)< [byte](#) > > &image, const [RasterFile-Format](#) ft=Auto, const int framenum=-1)

*Write a frame to disk.*

- void `writeGray` (const `Image< byte >` &image, const `RasterFileFormat` ft, const int framenum, const char \*otherstem,...)

*Write a frame to disk.*

- void `writeGray` (const `Image< byte >` &image, const `RasterFileFormat` ft=Auto, const int framenum=-1)

*Write a frame to disk.*

- void `writeFloat` (const `Image< float >` &image, const `RasterFileFormat` ft, const int framenum, const bool normalize, const char \*otherstem,...)

*Write a frame to disk.*

- void `writeFloat` (const `Image< float >` &image, const `RasterFileFormat` ft=Auto, const int framenum=-1, const bool normalize=true)

*Write a frame to disk.*

### 10.199.1 Detailed Description

A `FrameSeries` with additional facilities for writing output frames.

Definition at line 250 of file `FrameSeries.H`.

### 10.199.2 Constructor & Destructor Documentation

#### 10.199.2.1 `OutputFrameSeries::OutputFrameSeries (ModelManager & mgr, const std::string & descrName = "Frame Series", const std::string & tagName = "FrameSeries")`

Constructor.

Definition at line 457 of file `FrameSeries.C`.

#### 10.199.2.2 `OutputFrameSeries::~OutputFrameSeries ()` [virtual]

Destructor.

Definition at line 464 of file `FrameSeries.C`.

### 10.199.3 Member Function Documentation

**10.199.3.1** `void OutputFrameSeries::writeFloat (const Image< float > & image, const RasterFileFormat ft = Auto, const int framenum = -1, const bool normalize = true)`

Write a frame to disk.

Definition at line 560 of file FrameSeries.C.

References `image`, and `writeFloat()`.

**10.199.3.2** `void OutputFrameSeries::writeFloat (const Image< float > & image, const RasterFileFormat ft, const int framenum, const bool normalize, const char * otherstem, ...)`

Write a frame to disk.

Definition at line 532 of file FrameSeries.C.

References `FrameSeries::computeFileName()`, `Raster::Display()`, `ModelParam< RasterFileFormat >::getVal()`, `ModelParam< FrameRange >::getVal()`, `image`, `FrameRange::isMultiframe()`, `FrameSeries::okToDisplay()`, `RasterFileFormat`, `FrameSeries::resizeImage()`, `ModelManager::textLog()`, and `Raster::WriteFloat()`.

Referenced by `writeFloat()`.

**10.199.3.3** `void OutputFrameSeries::writeGray (const Image< byte > & image, const RasterFileFormat ft = Auto, const int framenum = -1)`

Write a frame to disk.

Definition at line 526 of file FrameSeries.C.

References `image`, and `writeGray()`.

**10.199.3.4** `void OutputFrameSeries::writeGray (const Image< byte > & image, const RasterFileFormat ft, const int framenum, const char * otherstem, ...)`

Write a frame to disk.

Definition at line 499 of file FrameSeries.C.

References `FrameSeries::computeFileName()`, `Raster::Display()`, `ModelParam< RasterFileFormat >::getVal()`, `ModelParam< FrameRange >::getVal()`, `image`, `FrameRange::isMultiframe()`, `FrameSeries::okToDisplay()`, `RasterFileFormat`, `FrameSeries::resizeImage()`, `ModelManager::textLog()`, and `Raster::WriteGray()`.



Referenced by writeGray().

**10.199.3.5** void OutputFrameSeries::writeRGB (const [Image](#)< [PixRGB](#)< byte > > & *image*, const [RasterFileFormat](#) *ft* = Auto, const int *framenum* = -1)

Write a frame to disk.

Definition at line 493 of file FrameSeries.C.

References [image](#), and [writeRGB\(\)](#).

**10.199.3.6** void OutputFrameSeries::writeRGB (const [Image](#)< [PixRGB](#)< byte > > & *image*, const [RasterFileFormat](#) *ft*, const int *framenum*, const char \* *otherstem*, ...)

Write a frame to disk.

Definition at line 468 of file FrameSeries.C.

References [FrameSeries::computeFileName\(\)](#), [Raster::Display\(\)](#), [ModelParam](#)< [RasterFileFormat](#) >::getVal(), [ModelParam](#)< [FrameRange](#) >::getVal(), [image](#), [FrameRange::isMultiframe\(\)](#), [FrameSeries::okToDisplay\(\)](#), [RasterFileFormat](#), [FrameSeries::resizeImage\(\)](#), and [Raster::WriteRGB\(\)](#).

Referenced by [writeRGB\(\)](#).

The documentation for this class was generated from the following files:

- [FrameSeries.H](#)
- [FrameSeries.C](#)

## 10.200 ParamMap Class Reference

Allows reading and writing of parameters to and from files or streams.

```
#include <ParamMap.H>
```

### Public Types

- enum [ReturnCode](#) { [MISSING](#), [UNCHANGED](#), [CHANGED](#) }  
*Return codes when getting params from the map.*

### Public Member Functions

- [ParamMap](#) ()  
*Default construct with an empty map.*
- [ParamMap](#) (const char \*fname)  
*Construct and [load\(\)](#) from fname.*
- [~ParamMap](#) ()  
*Destructor.*
- key\_iterator [keys\\_begin](#) () const  
*Get an iterator to the beginning of the keys sequence.*
- key\_iterator [keys\\_end](#) () const  
*Get an iterator to one-past-the-end of the keys sequence.*
- void [load](#) (std::istream &istrm)  
*Load in a parameter map from the given input stream.*
- void [load](#) (const char \*fname)  
*Convenience overload of the other [load\(\)](#).*
- void [format](#) (std::ostream &ostrm, int indentlev=0) const  
*Write a formatted parameter map to the given output stream.*
- void [format](#) (const char \*fname) const  
*Convenience overload of the other [format\(\)](#).*
- bool [hasParam](#) (const char \*paramname) const

*Query whether the parameter map has a particular named parameter.*

- bool `isLeaf` (const char \*paramname) const  
*Query whether the parameter map is a leaf or not.*
- void `print` (const char \*name) const  
*For testing/debugging only.*
- void `getAllSubmaps` (const char \*prefix, std::vector< SharedPtr< ParamMap > > &submaps, int &start\_ctr, int &start\_surr, int &end\_ctr, int &end\_surr)  
*Returns all submaps that have this prefix.*
- void `setAllSubmaps` (const char \*prefix, int &idx, std::vector< SharedPtr< ParamMap > > &submaps)  
*sets all submaps that have this prefix to the given values*

### Functions to extract named values from the ParamMap

There are several kinds of functions that differ in how they handle named parameters that are not found in the `ParamMap`. The first kind look for a named parameter, and trigger an error if that parameter is not found. The second kind take a default value as an additional argument, and return that value if the named parameter is not found. The third kind take a destination by reference, and leave that destination unchanged if the parameter is not found, and additionally return a `ReturnCode` specifying whether the parameter was missing, found-and-changed, or found-and-unchanged.

- SharedPtr< ParamMap > `getSubmap` (const char \*paramname) const  
*Get a named parameter as a nested parameter map.*
- const char \* `getStringParam` (const char \*paramname) const  
*Get a named parameter as a string value.*
- double `getDoubleParam` (const char \*paramname) const  
*Get a named parameter as a 'double' value.*
- int `getIntParam` (const char \*paramname) const  
*Get a named parameter as an 'int' value.*
- const char \* `getStringParam` (const char \*paramname, const char \*defval) const  
*Get a named parameter as a string value, or return the given default value.*
- double `getDoubleParam` (const char \*paramname, const double defval) const

*Get a named parameter as a 'double' value, or return the given default value.*

- `int getIntParam (const char *paramname, const int defval) const`  
*Get a named parameter as an 'int' value, or return the given default value.*
- `SharedPtr< ParamMap > lookupSubmap (const char *paramname)`  
*Returns the submap for the given name.*
- `ReturnCode queryStringParam (const char *paramname, const char *&result) const`  
*Get the named parameter, or do nothing if the parameter is missing.*
- `ReturnCode queryDoubleParam (const char *paramname, double &result) const`  
*Get the named parameter, or do nothing if the parameter is missing.*
- `ReturnCode queryIntParam (const char *paramname, int &result) const`  
*Get the named parameter, or do nothing if the parameter is missing.*

### Functions to place named values into the ParamMap

- `void putSubmap (const char *paramname, const SharedPtr< ParamMap > &val)`  
*Add a nested parameter map with a given name.*
- `void putStringParam (const char *paramname, const char *val)`  
*Add a string value with a given name.*
- `void putDoubleParam (const char *paramname, double val)`  
*Add a double-precision floating point value with a given name.*
- `void putIntParam (const char *paramname, int val)`  
*Add an integral value with a given name.*
- `void replaceStringParam (const char *paramname, const char *val)`  
*Replace string-param.*
- `void replaceDoubleParam (const char *paramname, double val)`  
*Replace double-param.*
- `void replaceIntParam (const char *paramname, int val)`  
*Replace int-param.*
- `void clear ()`  
*Remove all params from the ParamMap.*

## Static Public Member Functions

- [SharedPtr](#)< [ParamMap](#) > [loadPmapFile](#) (const char \*fname)  
*Convenience function to create a new ParamMap from a .pmap file.*
- [SharedPtr](#)< [ParamMap](#) > [loadConfFile](#) (const char \*fname)  
*Compatibility function to create a new ParamMap from a .conf file.*

### 10.200.1 Detailed Description

Allows reading and writing of parameters to and from files or streams.

ParamMap allows reading and writing of arbitrarily nested data structures to and from files or other streams.

Definition at line 47 of file ParamMap.H.

### 10.200.2 Member Enumeration Documentation

#### 10.200.2.1 enum [ParamMap::ReturnCode](#)

Return codes when getting params from the map.

**Enumeration values:**

***MISSING*** the param was not found

***UNCHANGED*** the param was found, but the value was unchanged

***CHANGED*** the param was found and its value was changed

Definition at line 64 of file ParamMap.H.

### 10.200.3 Constructor & Destructor Documentation

#### 10.200.3.1 [ParamMap::ParamMap](#) ()

Default construct with an empty map.

Definition at line 320 of file ParamMap.C.

#### 10.200.3.2 [ParamMap::ParamMap](#) (const char \* *fname*)

Construct and [load\(\)](#) from *fname*.

Definition at line 325 of file ParamMap.C.

References `load()`.

### 10.200.3.3 ParamMap::~~ParamMap ()

Destructor.

Definition at line 332 of file ParamMap.C.

## 10.200.4 Member Function Documentation

### 10.200.4.1 void ParamMap::clear ()

Remove all params from the ParamMap.

Definition at line 709 of file ParamMap.C.

### 10.200.4.2 void ParamMap::format (const char \* *fname*) const

Convenience overload of the other `format()`.

In this case we just build an ostream internally from the given filename, and then `format()` from that.

Definition at line 471 of file ParamMap.C.

References `format()`.

### 10.200.4.3 void ParamMap::format (std::ostream & *ostrm*, int *indentlev* = 0) const

Write a formatted parameter map to the given output stream.

The `indentlev` parameter is only used internally by ParamMap, and can be safely ignored by external callers.

Definition at line 456 of file ParamMap.C.

Referenced by `format()`, `ModelManager::saveConfig()`, `SingleChannelBeo::sendInput()`, and `VisualCortex::writeParamMap()`.

### 10.200.4.4 void ParamMap::getAllSubmaps (const char \* *prefix*, std::vector< SharedPtr< ParamMap > > & *submaps*, int & *start\_ctr*, int & *start\_surr*, int & *end\_ctr*, int & *end\_surr*)

Returns all submaps that have this prefix.

Definition at line 730 of file ParamMap.C.

References `getSubmap()`.

#### 10.200.4.5 **double ParamMap::getDoubleParam (const char \* *paramname*, const double *defval*) const**

Get a named parameter as a 'double' value, or return the given default value.

The default value will be returned if the named parameter is not found.

Definition at line 561 of file ParamMap.C.

References `queryDoubleParam()`.

#### 10.200.4.6 **double ParamMap::getDoubleParam (const char \* *paramname*) const**

Get a named parameter as a 'double' value.

Triggers an error if there is no parameter with the given name.

Definition at line 528 of file ParamMap.C.

Referenced by `SingleChannel::readFrom()`.

#### 10.200.4.7 **int ParamMap::getIntParam (const char \* *paramname*, const int *defval*) const**

Get a named parameter as an 'int' value, or return the given default value.

The default value will be returned if the named parameter is not found.

Definition at line 570 of file ParamMap.C.

References `queryIntParam()`.

#### 10.200.4.8 **int ParamMap::getIntParam (const char \* *paramname*) const**

Get a named parameter as an 'int' value.

Triggers an error if there is no parameter with the given name.

Definition at line 540 of file ParamMap.C.

#### 10.200.4.9 **const char \* ParamMap::getStringParam (const char \* *paramname*, const char \* *defval*) const**

Get a named parameter as a string value, or return the given default value.

The default value will be returned if the named parameter is not found.

Definition at line 552 of file ParamMap.C.

References `queryStringParam()`.

#### **10.200.4.10 `const char * ParamMap::getStringParam (const char * paramname) const`**

Get a named parameter as a string value.

Triggers an error if there is no parameter with the given name.

Definition at line 516 of file ParamMap.C.

#### **10.200.4.11 `SharedPtr< ParamMap > ParamMap::getSubmap (const char * paramname) const`**

Get a named parameter as a nested parameter map.

Triggers an error if there is no parameter with the given name.

Definition at line 504 of file ParamMap.C.

Referenced by `VisualCortex::findLikelihood()`, `SingleChannel::findLikelihood()`, `ComplexChannel::findLikelihood()`, `getAllSubmaps()`, `lookupSubmap()`, `VisualCortex::readFrom()`, `SingleChannel::readFrom()`, `ComplexChannel::readFrom()`, `ModelComponent::readParamsFrom()`, `VisualCortex::reco()`, `SingleChannel::reco()`, `ComplexChannel::reco()`, and `setAllSubmaps()`.

#### **10.200.4.12 `bool ParamMap::hasParam (const char * paramname) const`**

Query whether the parameter map has a particular named parameter.

Definition at line 487 of file ParamMap.C.

Referenced by `VisualCortex::findLikelihood()`, `ComplexChannel::findLikelihood()`, `lookupSubmap()`, `VisualCortex::readFrom()`, `SingleChannel::readFrom()`, `ComplexChannel::readFrom()`, `ModelComponent::readParamsFrom()`, `VisualCortex::reco()`, and `ComplexChannel::reco()`.

#### **10.200.4.13 `bool ParamMap::isLeaf (const char * paramname) const`**

Query whether the parameter map is a leaf or not.

Definition at line 494 of file ParamMap.C.

Referenced by `SingleChannel::readFrom()`.



**10.200.4.14 ParamMap::key\_iterator ParamMap::keys\_begin () const**

Get an iterator to the beginning of the keys sequence.

Definition at line 339 of file ParamMap.C.

**10.200.4.15 ParamMap::key\_iterator ParamMap::keys\_end () const**

Get an iterator to one-past-the-end of the keys sequence.

Definition at line 348 of file ParamMap.C.

**10.200.4.16 void ParamMap::load (const char \* *fname*)**

Convenience overload of the other [load\(\)](#).

In this case we just build an istream internally from the given filename, and then [load\(\)](#) from that.

Definition at line 440 of file ParamMap.C.

References [load\(\)](#).

**10.200.4.17 void ParamMap::load (std::istream & *istrm*)**

Load in a parameter map from the given input stream.

Definition at line 356 of file ParamMap.C.

Referenced by [SingleChannelBeoServer::check\(\)](#), [load\(\)](#), [ModelManager::loadConfig\(\)](#), [ParamMap\(\)](#), [ModelManager::parseCommandLine\(\)](#), and [VisualCortex::readParamMap\(\)](#).

**10.200.4.18 [SharedPtr< ParamMap >](#) ParamMap::loadConfFile (const char \* *fname*) [static]**

Compatibility function to create a new ParamMap from a .conf file.

See [readConfig](#) for details on the format of a .conf file. Basically, it is similar to the .pmap format, except that comments must be bracketed by whitespace-separated '#' characters on both sides, and that there is no nesting of data structures.

Definition at line 282 of file ParamMap.C.

**10.200.4.19** [SharedPtr](#)< [ParamMap](#) > [ParamMap::loadPmapFile](#) (const char \* *fname*) [static]

Convenience function to create a new ParamMap from a .pmap file.

Definition at line 274 of file ParamMap.C.

**10.200.4.20** [SharedPtr](#)< [ParamMap](#) > [ParamMap::lookupSubmap](#) (const char \* *paramname*)

Returns the submap for the given name.

If a submap for the given name exists, a [SharedPtr](#) to it is returned. If no submap for the given name has yet been created, one is created, and a [SharedPtr](#) to it is returned. This function differs from [getSubmap\(\)](#), which only returns existing submaps.

Definition at line 579 of file ParamMap.C.

References [getSubmap\(\)](#), [hasParam\(\)](#), and [putSubmap\(\)](#).

**10.200.4.21** [void](#) [ParamMap::print](#) (const char \* *name*) const

For testing/debugging only.

Prints the raw contents of this parameter map to standard output.

Definition at line 715 of file ParamMap.C.

**10.200.4.22** [void](#) [ParamMap::putDoubleParam](#) (const char \* *paramname*, double *val*)

Add a double-precision floating point value with a given name.

Definition at line 675 of file ParamMap.C.

Referenced by [GaborChannel::writeTo\(\)](#), and [ChannelBase::writeTo\(\)](#).

**10.200.4.23** [void](#) [ParamMap::putIntParam](#) (const char \* *paramname*, int *val*)

Add an integral value with a given name.

Definition at line 681 of file ParamMap.C.

Referenced by [GaborChannel::writeTo\(\)](#).

**10.200.4.24 void ParamMap::putStringParam (const char \* *paramname*, const char \* *val*)**

Add a string value with a given name.

Definition at line 668 of file ParamMap.C.

Referenced by ModelParamBase::writeTo().

**10.200.4.25 void ParamMap::putSubmap (const char \* *paramname*, const SharedPtr< ParamMap > & *val*)**

Add a nested parameter map with a given name.

Definition at line 662 of file ParamMap.C.

Referenced by lookupSubmap(), setAllSubmaps(), ModelComponent::writeParamsTo(), VisualCortex::writeTo(), SingleChannel::writeTo(), and ComplexChannel::writeTo().

**10.200.4.26 ParamMap::ReturnCode ParamMap::queryDoubleParam (const char \* *paramname*, double & *result*) const**

Get the named parameter, or do nothing if the parameter is missing.

The ReturnCode describes what happened when looking up the param.

Definition at line 617 of file ParamMap.C.

References CHANGED, MISSING, and UNCHANGED.

Referenced by getDoubleParam(), GaborChannel::readFrom(), and ChannelBase::readFrom().

**10.200.4.27 ParamMap::ReturnCode ParamMap::queryIntParam (const char \* *paramname*, int & *result*) const**

Get the named parameter, or do nothing if the parameter is missing.

The ReturnCode describes what happened when looking up the param.

Definition at line 640 of file ParamMap.C.

References CHANGED, MISSING, and UNCHANGED.

Referenced by getIntParam(), and GaborChannel::readFrom().

**10.200.4.28** [ParamMap::ReturnCode](#) [ParamMap::queryStringParam](#) (const char \* *paramname*, const char \*& *result*) const

Get the named parameter, or do nothing if the parameter is missing.

The ReturnCode describes what happened when looking up the param.

Definition at line 592 of file ParamMap.C.

References ASSERT, CHANGED, MISSING, and UNCHANGED.

Referenced by getStringParam(), and ModelParamBase::readFrom().

**10.200.4.29** [void ParamMap::replaceDoubleParam](#) (const char \* *paramname*, double *val*)

Replace double-param.

Definition at line 695 of file ParamMap.C.

**10.200.4.30** [void ParamMap::replaceIntParam](#) (const char \* *paramname*, int *val*)

Replace int-param.

Definition at line 702 of file ParamMap.C.

**10.200.4.31** [void ParamMap::replaceStringParam](#) (const char \* *paramname*, const char \* *val*)

Replace string-param.

Definition at line 687 of file ParamMap.C.

**10.200.4.32** [void ParamMap::setAllSubmaps](#) (const char \* *prefix*, int & *idx*, [std::vector](#)< [SharedPtr](#)< [ParamMap](#) > > & *submaps*)

sets all submaps that have this prefix to the given values

Definition at line 782 of file ParamMap.C.

References getSubmap(), and putSubmap().

The documentation for this class was generated from the following files:

- [ParamMap.H](#)
- [ParamMap.C](#)

## 10.201 ParamMap::Error Class Reference

Exception class thrown by [ParamMap](#) functions in case of error.

```
#include <ParamMap.H>
```

### 10.201.1 Detailed Description

Exception class thrown by [ParamMap](#) functions in case of error.

An exception class that will be thrown by ParamMap's functions in case of error.

Definition at line 83 of file ParamMap.H.

The documentation for this class was generated from the following file:

- [ParamMap.H](#)

## 10.202 PbmParser Class Reference

This class handles reading/parsing of PGM/PPM image files.

```
#include <PbmParser.H>
```

### Public Member Functions

- [PbmParser](#) (const char \*fname)  
*Initialize with a PGM or PPM file (either is OK).*
- [~PbmParser](#) ()  
*Destructor.*
- bool [isBitmap](#) () const  
*Query whether the file is a black+white bitmap.*
- bool [isGray](#) () const  
*Query whether the file is grayscale.*
- bool [isRGB](#) () const  
*Query whether the file is RGB.*
- [Dims](#) [getDims](#) () const  
*Get the dims of the image without fully reading it.*
- [std::string](#) [getComments](#) () const  
*Get the comments from the image without fully reading it.*
- [Image](#)< [PixRGB](#)< byte > > [getRGB](#) ()  
*Get an RGB image from the file.*
- [Image](#)< byte > [getGray](#) ()  
*Get a grayscale image from the file.*
- [Image](#)< float > [getFloat](#) ()  
*Get a float grayscale image from the file.*

### 10.202.1 Detailed Description

This class handles reading/parsing of PGM/PPM image files.

Currently, pbm modes 2, 3, 5, and 6 are supported (i.e., ascii or raw, grayscale or RGB). Modes 1 and 4 (for 1-bit black+white bitmaps) are **not** currently supported.

Definition at line 53 of file PbmParser.H.

### 10.202.2 Constructor & Destructor Documentation

#### 10.202.2.1 PbmParser::PbmParser (const char \* *fname*)

Initialize with a PGM or PPM file (either is OK).

Definition at line 62 of file PbmParser.C.

#### 10.202.2.2 PbmParser::~PbmParser ()

Destructor.

Definition at line 105 of file PbmParser.C.

### 10.202.3 Member Function Documentation

#### 10.202.3.1 std::string PbmParser::getComments () const

Get the comments from the image without fully reading it.

Definition at line 127 of file PbmParser.C.

Referenced by Raster::getImageMeta().

#### 10.202.3.2 Dims PbmParser::getDims () const

Get the dims of the image without fully reading it.

Definition at line 123 of file PbmParser.C.

Referenced by Raster::getImageDims().

#### 10.202.3.3 Image< float > PbmParser::getFloat ()

Get a float grayscale image from the file.

This works regardless of the actual mode of the underlying file (i.e., if the file was RGB, `getFloat()` the returns luminance of that).

Definition at line 147 of file PbmParser.C.

References `isBitmap()`, `isGray()`, and `luminance()`.

#### **10.202.3.4   Image< byte > PbmParser::getGray ()**

Get a grayscale image from the file.

This works regardless of the actual mode of the underlying file (i.e., if the file was RGB, `getGray()` the returns luminance of that).

Definition at line 139 of file PbmParser.C.

References `isBitmap()`, `isGray()`, and `luminance()`.

#### **10.202.3.5   Image< PixRGB< byte > > PbmParser::getRGB ()**

Get an RGB image from the file.

This works regardless of the actual mode of the underlying file (i.e., if the file was grayscale, this will convert that to RGB).

Definition at line 131 of file PbmParser.C.

References `isBitmap()`, `isGray()`, and `toRGB()`.

#### **10.202.3.6   bool PbmParser::isBitmap () const**

Query whether the file is a black+white bitmap.

Definition at line 111 of file PbmParser.C.

Referenced by `getFloat()`, `getGray()`, and `getRGB()`.

#### **10.202.3.7   bool PbmParser::isGray () const**

Query whether the file is grayscale.

Definition at line 115 of file PbmParser.C.

Referenced by `getFloat()`, `getGray()`, and `getRGB()`.

#### **10.202.3.8   bool PbmParser::isRGB () const**

Query whether the file is RGB.



Definition at line 119 of file PbmParser.C.

The documentation for this class was generated from the following files:

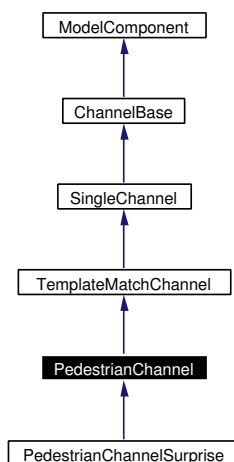
- [PbmParser.H](#)
- [PbmParser.C](#)

## 10.203 PedestrianChannel Class Reference

A template-matching channel for detecting pedestrians.

```
#include <Channels.H>
```

Inheritance diagram for PedestrianChannel:



Collaboration diagram for PedestrianChannel:

### Public Member Functions

- [PedestrianChannel](#) ([ModelManager](#) &mgr)  
*Constructor.*
- virtual [~PedestrianChannel](#) ()  
*Virtual destructor.*

### Protected Member Functions

- virtual void [start1](#) ()

### Protected Attributes

- [ModelParam](#)< int > [itsMinScale](#)

*smallest scale to use in template matching*

- [ModelParam](#)< int > [itsMaxScale](#)

*largest scale to use in template matching*

### 10.203.1 Detailed Description

A template-matching channel for detecting pedestrians.

Definition at line 405 of file Channels.H.

### 10.203.2 Constructor & Destructor Documentation

#### 10.203.2.1 PedestrianChannel::PedestrianChannel ([ModelManager](#) & mgr)

Constructor.

Definition at line 1653 of file Channels.C.

#### 10.203.2.2 PedestrianChannel::~~PedestrianChannel () [virtual]

Virtual destructor.

Definition at line 1660 of file Channels.C.

### 10.203.3 Member Function Documentation

#### 10.203.3.1 void PedestrianChannel::start1 () [protected, virtual]

If you overload this, make sure you call [SingleChannel::start1\(\)](#) at the beginning of your overload

Reimplemented from [TemplateMatchChannel](#).

Definition at line 1664 of file Channels.C.

References [ModelParam](#)< int >::getVal(), [itsMaxScale](#), [itsMinScale](#), [ModelParam](#)< [LevelSpec](#) >::setVal(), and [TemplateMatchChannel::start1\(\)](#).

## 10.203.4 Member Data Documentation

### 10.203.4.1 [ModelParam<int>](#) [PedestrianChannel::itsMaxScale](#) [protected]

largest scale to use in template matching

Definition at line 416 of file Channels.H.

Referenced by [start1\(\)](#).

### 10.203.4.2 [ModelParam<int>](#) [PedestrianChannel::itsMinScale](#) [protected]

smallest scale to use in template matching

Definition at line 415 of file Channels.H.

Referenced by [start1\(\)](#).

The documentation for this class was generated from the following files:

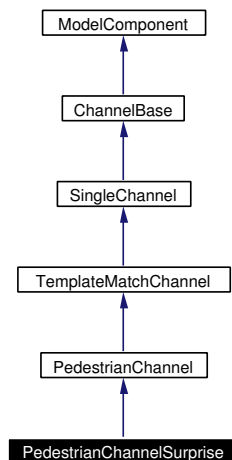
- [Channels.H](#)
- [Channels.C](#)

## 10.204 PedestrianChannelSurprise Class Reference

A template-matching channel for detecting pedestrians.

```
#include <ChannelsSurprise.H>
```

Inheritance diagram for PedestrianChannelSurprise:



Collaboration diagram for PedestrianChannelSurprise:

### Public Member Functions

- [PedestrianChannelSurprise](#) ([ModelManager](#) &mgr)  
*Constructor.*
- virtual [~PedestrianChannelSurprise](#) ()  
*Virtual destructor.*
- virtual [Image](#)< float > [computeSubmap](#) (const [uint](#) index) const  
*[computeSubmap\(\)](#) is special and will return surprise*

### Protected Attributes

- [SharedPtr](#)< [SingleChannelSurprise](#) > itsSCS  
*our surprise computer*

### 10.204.1 Detailed Description

A template-matching channel for detecting pedestrians.

Definition at line 359 of file ChannelsSurprise.H.

### 10.204.2 Constructor & Destructor Documentation

#### 10.204.2.1 PedestrianChannelSurprise::PedestrianChannelSurprise ([ModelManager](#) & mgr)

Constructor.

Definition at line 327 of file ChannelsSurprise.C.

References [ModelComponent::addSubComponent\(\)](#).

#### 10.204.2.2 PedestrianChannelSurprise::~~PedestrianChannelSurprise () [virtual]

Virtual destructor.

Definition at line 333 of file ChannelsSurprise.C.

### 10.204.3 Member Function Documentation

#### 10.204.3.1 [Image](#)< float > PedestrianChannelSurprise::computeSubmap (const [uint](#) index) const [virtual]

[computeSubmap\(\)](#) is special and will return surprise

Reimplemented from [SingleChannel](#).

Definition at line 336 of file ChannelsSurprise.C.

References [uint](#).

### 10.204.4 Member Data Documentation

#### 10.204.4.1 [SharedPtr](#)<[SingleChannelSurprise](#)> [PedestrianChannelSurprise::itsSCS](#) [protected]

our surprise computer

Definition at line 372 of file ChannelsSurprise.H.

The documentation for this class was generated from the following files:

- [ChannelsSurprise.H](#)
- ChannelsSurprise.C

## 10.205 PfmParser Class Reference

This class handles reading/parsing of PFM image files.

```
#include <PfmParser.H>
```

### Public Member Functions

- [PfmParser](#) (const char \*fname)  
*Initialize with a PGM or PPM file (either is OK).*
- [~PfmParser](#) ()  
*Destructor.*
- bool [isBitmap](#) () const  
*Query whether the file is a black+white bitmap.*
- bool [isGray](#) () const  
*Query whether the file is grayscale.*
- bool [isRGB](#) () const  
*Query whether the file is RGB.*
- [Dims](#) [getDims](#) () const  
*Get the dims of the image without fully reading it.*
- [std::string](#) [getComments](#) () const  
*Get the comments from the image without fully reading it.*
- [Image](#)< [PixRGB](#)< [byte](#) > > [getRGB](#) ()  
*Get an RGB image from the file.*
- [Image](#)< [byte](#) > [getGray](#) ()  
*Get a grayscale image from the file.*
- [Image](#)< [float](#) > [getFloat](#) ()  
*Get a float grayscale image from the file.*



### 10.205.1 Detailed Description

This class handles reading/parsing of PFM image files.

PFM is an unofficial format that looks very much like PGM except that raw pixel values are floats. Caution, this is not portable across endianness. The PNM magic for this unofficial format is PF.

Definition at line 54 of file PfmParser.H.

### 10.205.2 Constructor & Destructor Documentation

#### 10.205.2.1 PfmParser::PfmParser (const char \* *fname*)

Initialize with a PGM or PPM file (either is OK).

Definition at line 62 of file PfmParser.C.

#### 10.205.2.2 PfmParser::~PfmParser ()

Destructor.

Definition at line 105 of file PfmParser.C.

### 10.205.3 Member Function Documentation

#### 10.205.3.1 std::string PfmParser::getComments () const

Get the comments from the image without fully reading it.

Definition at line 127 of file PfmParser.C.

#### 10.205.3.2 Dims PfmParser::getDims () const

Get the dims of the image without fully reading it.

Definition at line 123 of file PfmParser.C.

#### 10.205.3.3 Image< float > PfmParser::getFloat ()

Get a float grayscale image from the file.

This works regardless of the actual mode of the underlying file (i.e., if the file was RGB, `getFloat()` the returns luminance of that).

Definition at line 139 of file PfmParser.C.

**10.205.3.4   [Image](#)< [byte](#) > PfmParser::getGray ()**

Get a grayscale image from the file.

This works regardless of the actual mode of the underlying file (i.e., if the file was RGB, [getGray\(\)](#) the returns luminance of that).

Definition at line 135 of file PfmParser.C.

**10.205.3.5   [Image](#)< [PixRGB](#)< [byte](#) > > PfmParser::getRGB ()**

Get an RGB image from the file.

This works regardless of the actual mode of the underlying file (i.e., if the file was grayscale, this will convert that to RGB).

Definition at line 131 of file PfmParser.C.

References toRGB().

**10.205.3.6   [bool](#) PfmParser::isBitmap () const**

Query whether the file is a black+white bitmap.

Definition at line 111 of file PfmParser.C.

**10.205.3.7   [bool](#) PfmParser::isGray () const**

Query whether the file is grayscale.

Definition at line 115 of file PfmParser.C.

**10.205.3.8   [bool](#) PfmParser::isRGB () const**

Query whether the file is RGB.

Definition at line 119 of file PfmParser.C.

The documentation for this class was generated from the following files:

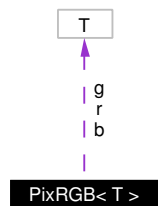
- [PfmParser.H](#)
- [PfmParser.C](#)

## 10.206 PixRGB< T > Class Template Reference

Generic pixel class for color RGB pixels.

```
#include <Pixels.H>
```

Collaboration diagram for PixRGB< T >:



### Public Member Functions

- [PixRGB \(\)](#)  
*Construct an RGB pixel, black.*
- [PixRGB \(const T redval, const T greenval, const T blueval\)](#)  
*Construct an RGB pixel with given values red, green, blue.*
- [template<class T2, class T3, class T4> PixRGB \(const T2 redval, const T3 greenval, const T4 blueval\)](#)  
*Construct from given values, clamping & converting as needed.*
- [PixRGB \(const T lum\)](#)  
*Construct an RGB pixel of luminance lum.*
- [template<class T2> PixRGB \(const T2 lum\)](#)  
*Construct an RGB pixel of luminance lum, clamping & converting as needed.*
- [PixRGB \(const PixRGB< T > &A\)](#)  
*Construct an RGB pixel from another.*
- [template<class T2> PixRGB \(const PixRGB< T2 > &A\)](#)  
*Constructor from another PixRGB with possibly different base type.*
- [PixRGB< T > & operator= \(const PixRGB< T > &A\)](#)  
*Construct an RGB pixel equal to another.*

- `template<class T2> PixRGB< T > & operator= (const PixRGB< T2 > &A)`  
*Construct an RGB pixel equal to another of possibly different base type.*
- `T luminance () const`  
*Compute luminance as  $(r + g + b) / 3$ .*
- `T red () const`  
*Access red channel.*
- `T green () const`  
*Access green channel.*
- `T blue () const`  
*Access blue channel.*
- `template<class T2> void setRed (const T2 redval)`  
*Set red channel value, clamping & converting as necessary.*
- `template<class T2> void setGreen (const T2 greenval)`  
*Set green channel value, clamping & converting as necessary.*
- `template<class T2> void setBlue (const T2 blueval)`  
*Set blue channel value, clamping & converting as necessary.*
- `template<class T2, class T3, class T4> void set (const T2 redval, const T3 greenval, const T4 blueval)`  
*Set all channel values, clamping & converting as necessary.*
- `bool operator== (const PixRGB< T > &other) const`  
*PixRGB equality operator.*
- `bool operator!= (const PixRGB< T > &other) const`  
*PixRGB inequality operator.*
- `template<class T2> PixRGB< T > & operator+= (const T2 val)`  
*Add a value to all components, clamp result as necessary.*
- `template<class T2> PixRGB< T > & operator+= (const PixRGB< T2 > &A)`  
*Add component-by-component, clamp result as necessary.*
- `template<class T2> PixRGB< T > & operator-= (const T2 val)`

*PixRGB operator.*

- `template<class T2> PixRGB< T > & operator-= (const PixRGB< T2 > &A)`  
*PixRGB operator.*
- `template<class T2> PixRGB< T > & operator *= (const T2 val)`  
*PixRGB operator.*
- `template<class T2> PixRGB< T > & operator *= (const PixRGB< T2 > &A)`  
*PixRGB operator.*
- `template<class T2> PixRGB< T > & operator/= (const T2 val)`  
*PixRGB operator.*
- `template<class T2> PixRGB< T > & operator/= (const PixRGB< T2 > &A)`  
*PixRGB operator.*
- `template<class T2> PixRGB< typename promote_trait< T, T2 >::TP > operator+ (const PixRGB< T2 > &B) const`  
*Add component-by-component, promote & clamp as necessary.*
- `template<class T2> PixRGB< typename promote_trait< T, T2 >::TP > operator+ (const T2 val) const`  
*PixRGB operator.*
- `template<class T2> PixRGB< typename promote_trait< T, T2 >::TP > operator- (const PixRGB< T2 > &B) const`  
*PixRGB operator.*
- `template<class T2> PixRGB< typename promote_trait< T, T2 >::TP > operator- (const T2 val) const`  
*PixRGB operator.*
- `template<class T2> PixRGB< typename promote_trait< T, T2 >::TP > operator * (const PixRGB< T2 > &B) const`  
*PixRGB operator.*
- `template<class T2> PixRGB< typename promote_trait< T, T2 >::TP > operator * (const T2 val) const`  
*PixRGB operator.*

- `template<class T2> PixRGB< typename promote_trait< T, T2 >::TP > operator/ (const PixRGB< T2 > &B) const`  
*PixRGB operator.*
- `template<class T2> PixRGB< typename promote_trait< T, T2 >::TP > operator/ (const T2 val) const`  
*PixRGB operator.*
- `bool operator> (const PixRGB< T > &B) const`  
*This nonsense comparison operator will abort the program when called.*
- `bool operator< (const PixRGB< T > &B) const`  
*This nonsense comparison operator will abort the program when called.*
- `void getYIQ (T &y, T &i, T &q) const`  
*Color conversion for YIQ.*
- `void getYUV (T &y, T &u, T &v) const`  
*Color conversion for YUV.*
- `void getHSV (T &h, T &s, T &v) const`  
*Color conversion for HSV.*
- `void setYUV (const T y, const T u, const T v)`  
*Color conversion for YUV.*
- `void setHSV (const T h, const T s, const T v)`  
*Color conversion for HSV.*

### 10.206.1 Detailed Description

**template<class T> class PixRGB< T >**

Generic pixel class for color RGB pixels.

These definitions are for color pixels. The pixels initialize to black through the default constructor. The following other methods are required if those pixels are to be used for methods available for color images (see [Image.H](#)):

- `constructor(red, green, blue)` - construct from an RGB triplet
- `constructor(lum)` - construct gray pixel(lum, lum, lum)

- [luminance\(\)](#) - return luminance value
- [red\(\)](#), [green\(\)](#), [blue\(\)](#) - return respective component values
- [setRed\(\)](#), [setGreen\(\)](#), [setBlue\(\)](#) - set respective component values

WARNING: you should NOT define comparison operators for color pixels, otherwise you will get the impression that many [Image](#) methods work while they will actually be bogus...

Definition at line 74 of file Pixels.H.

## 10.206.2 Constructor & Destructor Documentation

### 10.206.2.1 `template<class T> PixRGB< T >::PixRGB ()` [inline]

Construct an RGB pixel, black.

Definition at line 251 of file Pixels.H.

### 10.206.2.2 `template<class T> PixRGB< T >::PixRGB (const T redval, const T greenval, const T blueval)` [inline]

Construct an RGB pixel with given values red, green, blue.

Definition at line 256 of file Pixels.H.

### 10.206.2.3 `template<class T> template<class T2, class T3, class T4> PixRGB< T >::PixRGB (const T2 redval, const T3 greenval, const T4 blueval)` [inline]

Construct from given values, clamping & converting as needed.

Definition at line 261 of file Pixels.H.

### 10.206.2.4 `template<class T> PixRGB< T >::PixRGB (const T lum)` [inline]

Construct an RGB pixel of luminance lum.

Definition at line 270 of file Pixels.H.

References [lum](#).

### 10.206.2.5 `template<class T> template<class T2> PixRGB< T >::PixRGB (const T2 lum) [inline]`

Construct an RGB pixel of luminance lum, clamping & converting as needed.

Definition at line 275 of file Pixels.H.

References lum.

### 10.206.2.6 `template<class T> PixRGB< T >::PixRGB (const PixRGB< T > & A) [inline]`

Construct an RGB pixel from another.

Definition at line 280 of file Pixels.H.

References [PixRGB](#)< T >::b, [PixRGB](#)< T >::g, and [PixRGB](#)< T >::r.

### 10.206.2.7 `template<class T> template<class T2> PixRGB< T >::PixRGB (const PixRGB< T2 > & A) [inline]`

Constructor from another [PixRGB](#) with possibly different base type.

Definition at line 285 of file Pixels.H.

References [PixRGB](#)< T >::blue(), [PixRGB](#)< T >::green(), and [PixRGB](#)< T >::red().

## 10.206.3 Member Function Documentation

### 10.206.3.1 `template<class T> T PixRGB< T >::blue () const [inline]`

Access blue channel.

Definition at line 324 of file Pixels.H.

Referenced by [convertToString\(\)](#), [isFinite\(\)](#), [PixRGB](#)< T >::operator \*(), [PixRGB](#)< T >::operator +(), [PixRGB](#)< T >::operator -(), [PixRGB](#)< T >::operator /(), [PixRGB](#)< T >::operator =(), [PixRGB](#)< T >::PixRGB(), [RGBImage2mexArray\(\)](#), [RGBtoCIE\(\)](#), and [sqrt\(\)](#).

### 10.206.3.2 `template<class T> void PixRGB< T >::getHSV (T & h, T & s, T & v) const [inline]`

Color conversion for HSV.

Definition at line 526 of file Pixels.H.



**10.206.3.3** `template<class T> void PixRGB< T >::getYIQ (T & y, T & i, T & q) const [inline]`

Color conversion for YIQ.

Definition at line 504 of file Pixels.H.

**10.206.3.4** `template<class T> void PixRGB< T >::getYUV (T & y, T & u, T & v) const [inline]`

Color conversion for YUV.

Definition at line 514 of file Pixels.H.

**10.206.3.5** `template<class T> T PixRGB< T >::green () const [inline]`

Access green channel.

Definition at line 319 of file Pixels.H.

Referenced by `convertToString()`, `isFinite()`, `PixRGB< T >::operator *=(())`, `PixRGB< T >::operator +=()`, `PixRGB< T >::operator -=()`, `PixRGB< T >::operator /=()`, `PixRGB< T >::operator =()`, `PixRGB< T >::PixRGB()`, `RGBImage2mexArray()`, `RGBtoCIE()`, and `sqrt()`.

**10.206.3.6** `template<class T> T PixRGB< T >::luminance () const [inline]`

Compute luminance as  $(r + g + b) / 3$ .

Definition at line 309 of file Pixels.H.

**10.206.3.7** `template<class T> template<class T2> PixRGB< typename promote_trait< T, T2 >::TP > PixRGB< T >::operator * (const T2 val) const [inline]`

PixRGB operator.

Definition at line 468 of file Pixels.H.

**10.206.3.8** `template<class T> template<class T2> PixRGB< typename promote_trait< T, T2 >::TP > PixRGB< T >::operator * (const PixRGB< T2 > & B) const [inline]`

PixRGB operator.

Definition at line 463 of file Pixels.H.

References B.

**10.206.3.9** `template<class T> template<class T2> PixRGB< T > & PixRGB< T >::operator *= (const PixRGB< T2 > & A) [inline]`

PixRGB operator.

Definition at line 413 of file Pixels.H.

References `PixRGB< T >::blue()`, `PixRGB< T >::green()`, and `PixRGB< T >::red()`.

**10.206.3.10** `template<class T> template<class T2> PixRGB< T > & PixRGB< T >::operator *= (const T2 val) [inline]`

PixRGB operator.

Definition at line 403 of file Pixels.H.

**10.206.3.11** `template<class T> bool PixRGB< T >::operator!= (const PixRGB< T > & other) const [inline]`

PixRGB inequality operator.

Definition at line 358 of file Pixels.H.

References `PixRGB< T >::b`, `PixRGB< T >::g`, and `PixRGB< T >::r`.

**10.206.3.12** `template<class T> template<class T2> PixRGB< typename promote_trait< T, T2 >::TP > PixRGB< T >::operator+ (const T2 val) const [inline]`

PixRGB operator.

Definition at line 448 of file Pixels.H.

**10.206.3.13** `template<class T> template<class T2> PixRGB< typename promote_trait< T, T2 >::TP > PixRGB< T >::operator+ (const PixRGB< T2 > & B) const [inline]`

Add component-by-component, promote & clamp as necessary.

Definition at line 443 of file Pixels.H.

References B.

**10.206.3.14** `template<class T> template<class T2> PixRGB< T > &  
 PixRGB< T >::operator+= (const PixRGB< T2 > & A)  
 [inline]`

Add component-by-component, clamp result as necessary.

Definition at line 373 of file Pixels.H.

References PixRGB< T >::blue(), PixRGB< T >::green(), and PixRGB< T >::red().

**10.206.3.15** `template<class T> template<class T2> PixRGB< T > &  
 PixRGB< T >::operator+= (const T2 val) [inline]`

Add a value to all components, clamp result as necessary.

Definition at line 363 of file Pixels.H.

**10.206.3.16** `template<class T> template<class T2> PixRGB< typename  
 promote_trait< T, T2 >::TP > PixRGB< T >::operator- (const T2  
 val) const [inline]`

PixRGB operator.

Definition at line 458 of file Pixels.H.

**10.206.3.17** `template<class T> template<class T2> PixRGB< typename  
 promote_trait< T, T2 >::TP > PixRGB< T >::operator- (const  
 PixRGB< T2 > & B) const [inline]`

PixRGB operator.

Definition at line 453 of file Pixels.H.

References B.

**10.206.3.18** `template<class T> template<class T2> PixRGB< T > &  
 PixRGB< T >::operator-= (const PixRGB< T2 > & A)  
 [inline]`

PixRGB operator.

Definition at line 393 of file Pixels.H.

References PixRGB< T >::blue(), PixRGB< T >::green(), and PixRGB< T >::red().

**10.206.3.19** `template<class T> template<class T2> PixRGB< T > &  
PixRGB< T >::operator-= (const T2 val) [inline]`

PixRGB operator.

Definition at line 383 of file Pixels.H.

**10.206.3.20** `template<class T> template<class T2> PixRGB< typename  
promote\_trait< T, T2 >::TP > PixRGB< T >::operator/ (const T2  
val) const [inline]`

PixRGB operator.

Definition at line 483 of file Pixels.H.

**10.206.3.21** `template<class T> template<class T2> PixRGB< typename  
promote\_trait< T, T2 >::TP > PixRGB< T >::operator/ (const  
PixRGB< T2 > & B) const [inline]`

PixRGB operator.

Definition at line 475 of file Pixels.H.

References B.

**10.206.3.22** `template<class T> template<class T2> PixRGB< T > &  
PixRGB< T >::operator/= (const PixRGB< T2 > & A)  
[inline]`

PixRGB operator.

Definition at line 433 of file Pixels.H.

References [PixRGB< T >::blue\(\)](#), [PixRGB< T >::green\(\)](#), and [PixRGB< T >::red\(\)](#).

**10.206.3.23** `template<class T> template<class T2> PixRGB< T > &  
PixRGB< T >::operator/= (const T2 val) [inline]`

PixRGB operator.

Definition at line 423 of file Pixels.H.

**10.206.3.24** `template<class T> bool PixRGB< T >::operator< (const  
PixRGB< T > & B) const [inline]`

This nonsense comparison operator will abort the program when called.

It is implemented only to keep the compiler happy by allowing it to instantiate some methods that use comparisons but should never be used on color pixels. For instance, this is necessary for [centerSurround\(\)](#) if we want to be able to use color pyramids.

Definition at line 496 of file Pixels.H.

**10.206.3.25** `template<class T> template<class T2> PixRGB< T > &  
PixRGB< T >::operator= (const PixRGB< T2 > & A)  
[inline]`

Construct an RGB pixel equal to another of possibly different base type.

Definition at line 299 of file Pixels.H.

References [PixRGB< T >::blue\(\)](#), [PixRGB< T >::green\(\)](#), and [PixRGB< T >::red\(\)](#).

**10.206.3.26** `template<class T> PixRGB< T > & PixRGB< T >::operator=  
(const PixRGB< T > & A) [inline]`

Construct an RGB pixel equal to another.

Definition at line 294 of file Pixels.H.

References [PixRGB< T >::b](#), [PixRGB< T >::g](#), and [PixRGB< T >::r](#).

**10.206.3.27** `template<class T> bool PixRGB< T >::operator== (const  
PixRGB< T > & other) const [inline]`

[PixRGB](#) equality operator.

Definition at line 353 of file Pixels.H.

References [PixRGB< T >::b](#), [PixRGB< T >::g](#), and [PixRGB< T >::r](#).

**10.206.3.28** `template<class T> bool PixRGB< T >::operator> (const  
PixRGB< T > & B) const [inline]`

This nonsense comparison operator will abort the program when called.

It is implemented only to keep the compiler happy by allowing it to instantiate some methods that use comparisons but should never be used on color pixels. For instance, this is necessary for [centerSurround\(\)](#) if we want to be able to use color pyramids.

Definition at line 488 of file Pixels.H.

**10.206.3.29** `template<class T> T PixRGB< T >::red () const` [inline]

Access red channel.

Definition at line 314 of file Pixels.H.

Referenced by `convertToString()`, `isFinite()`, `PixRGB< T >::operator *()`, `PixRGB< T >::operator +()`, `PixRGB< T >::operator -()`, `PixRGB< T >::operator /()`, `PixRGB< T >::operator =()`, `PixRGB< T >::PixRGB()`, `RGBImage2mexArray()`, `RGBtoCIE()`, and `sqrt()`.

**10.206.3.30** `template<class T> template<class T2, class T3, class T4> void PixRGB< T >::set (const T2 redval, const T3 greenval, const T4 blueval)` [inline]

Set all channel values, clamping & converting as necessary.

Definition at line 344 of file Pixels.H.

Referenced by `convertFromString()`, and `ImageSpring< T >::getClusteredImage()`.

**10.206.3.31** `template<class T> template<class T2> void PixRGB< T >::setBlue (const T2 blueval)` [inline]

Set blue channel value, clamping & converting as necessary.

Definition at line 339 of file Pixels.H.

Referenced by `convertFromString()`, `SimulationViewerEyeMvt2::eyeMovement()`, `SimulationViewerEyeMvt::eyeMovement()`, and `mexArray2RGBImage()`.

**10.206.3.32** `template<class T> template<class T2> void PixRGB< T >::setGreen (const T2 greenval)` [inline]

Set green channel value, clamping & converting as necessary.

Definition at line 334 of file Pixels.H.

Referenced by `convertFromString()`, `SimulationViewerEyeMvt2::eyeMovement()`, `SimulationViewerEyeMvt::eyeMovement()`, and `mexArray2RGBImage()`.

**10.206.3.33** `template<class T> void PixRGB< T >::setHSV (const T h, const T s, const T v)` [inline]

Color conversion for HSV.

Definition at line 563 of file Pixels.H.

**10.206.3.34** `template<class T> template<class T2> void PixRGB< T >::setRed (const T2 redval) [inline]`

Set red channel value, clamping & converting as necessary.

Definition at line 329 of file Pixels.H.

Referenced by `convertFromString()`, `SimulationViewerEyeMvt2::eyeMovement()`, `SimulationViewerEyeMvt::eyeMovement()`, and `mexArray2RGBImage()`.

**10.206.3.35** `template<class T> void PixRGB< T >::setYUV (const T y, const T u, const T v) [inline]`

Color conversion for YUV.

Definition at line 547 of file Pixels.H.

The documentation for this class was generated from the following file:

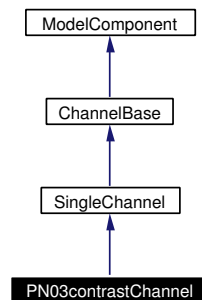
- [Pixels.H](#)

## 10.207 PN03contrastChannel Class Reference

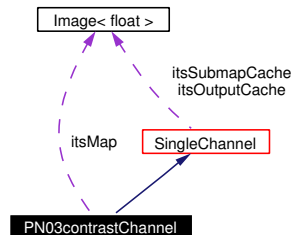
A Contrast channel (based on Parkhurst & Niebur 2003).

```
#include <Channels.H>
```

Inheritance diagram for PN03contrastChannel:



Collaboration diagram for PN03contrastChannel:



### Public Member Functions

- [PN03contrastChannel \(ModelManager &mgr\)](#)  
*Construct with standard params.*
- virtual [~PN03contrastChannel \(\)](#)  
*Destructor.*
- virtual bool [outputAvailable \(\)](#) const  
*overload*



## Protected Member Functions

- virtual void `doInput` (const `Image< PixRGB< float > > *`, const `Image< float > *bwimg`, const double `t`, const `Image< byte > &clipMask`)  
*special input function*
- virtual `Image< float > getOutput` ()  
*special output function*

## Protected Attributes

- `Image< float > itsMap`  
*our contrast map*

### 10.207.1 Detailed Description

A Contrast channel (based on Parkhurst & Niebur 2003).

CAUTION: this channel will throw a fatal exception if your `LevelSpec` contains more than one level or if that level is different from the `maplevel` or if the `deltas` are non-zero.

Definition at line 684 of file `Channels.H`.

### 10.207.2 Constructor & Destructor Documentation

#### 10.207.2.1 PN03contrastChannel::PN03contrastChannel (`ModelManager` & `mgr`)

Construct with standard params.

Definition at line 2940 of file `Channels.C`.

References `PN03CONTRAST`.

#### 10.207.2.2 PN03contrastChannel::~~PN03contrastChannel () [virtual]

Destructor.

Definition at line 2947 of file `Channels.C`.

### 10.207.3 Member Function Documentation

**10.207.3.1** `void PN03contrastChannel::doInput (const Image< PixRGB< float > > *, const Image< float > * bwimg, const double t, const Image< byte > & clipMask)` [protected, virtual]

special input function

Reimplemented from [SingleChannel](#).

Definition at line 2955 of file Channels.C.

References [ASSERT](#), [Image](#)< float >::beginw(), [LevelSpec](#)::delMax(), [LevelSpec](#)::delMin(), [Image](#)< T >::getHeight(), [Image](#)< T >::getVal(), [ModelParam](#)< [LevelSpec](#) >::getVal(), [Image](#)< T >::getWidth(), [LevelSpec](#)::levMax(), [LevelSpec](#)::levMin(), [LevelSpec](#)::mapLevel(), [Image](#)< float >::resize(), [sqrt](#)(), and [uint](#).

**10.207.3.2** `Image< float > PN03contrastChannel::getOutput ()` [protected, virtual]

special output function

Reimplemented from [SingleChannel](#).

Definition at line 3028 of file Channels.C.

**10.207.3.3** `bool PN03contrastChannel::outputAvailable () const` [virtual]

overload

Reimplemented from [SingleChannel](#).

Definition at line 2951 of file Channels.C.

References [Image](#)< float >::initialized().

### 10.207.4 Member Data Documentation

**10.207.4.1** `Image<float> PN03contrastChannel::itsMap` [protected]

our contrast map

Definition at line 705 of file Channels.H.

The documentation for this class was generated from the following files:

- [Channels.H](#)
- [Channels.C](#)

## 10.208 PngParser Class Reference

This class handles reading/parsing of PNG image files.

```
#include <PngParser.H>
```

### Public Member Functions

- [PngParser](#) (const char \*filename)  
*Intialize with the name of a PNG file.*
- [~PngParser](#) ()  
*Destructor.*
- bool [isGray](#) () const  
*Query whether the image file is grayscale.*
- bool [isColor](#) () const  
*Query whether the image file is color (RGB or color palette).*
- [Dims](#) [getDims](#) () const  
*Get the dims of the image without fully reading it.*
- [Image](#)< [PixRGB](#)< byte > > [getRGB](#) ()  
*Get an RGB image from the file.*
- [Image](#)< byte > [getGray](#) ()  
*Get a grayscale image from the file.*
- [Image](#)< float > [getFloat](#) ()  
*Get a float grayscale image from the file.*

### 10.208.1 Detailed Description

This class handles reading/parsing of PNG image files.

Currently, grayscale and RGB images are supported, with 8-bit or 16-bit pixel elements.

Definition at line 50 of file PngParser.H.

## 10.208.2 Constructor & Destructor Documentation

### 10.208.2.1 PngParser::PngParser (const char \* *filename*)

Intialize with the name of a PNG file.

Definition at line 68 of file PngParser.C.

References ASSERT.

### 10.208.2.2 PngParser::~PngParser ()

Destructor.

Definition at line 147 of file PngParser.C.

## 10.208.3 Member Function Documentation

### 10.208.3.1 Dims PngParser::getDims () const

Get the dims of the image without fully reading it.

Definition at line 193 of file PngParser.C.

Referenced by Raster::getImageDims().

### 10.208.3.2 Image< float > PngParser::getFloat ()

Get a float grayscale image from the file.

This works regardless of the actual mode of the underlying file (i.e., if the file was RGB, `getFloat()` returns the luminance of that).

Definition at line 207 of file PngParser.C.

References `isColor()`, `isGray()`, and `luminance()`.

### 10.208.3.3 Image< byte > PngParser::getGray ()

Get a grayscale image from the file.

This works regardless of the actual mode of the underlying file (i.e., if the file was RGB, `getGray()` returns the luminance of that).

Definition at line 197 of file PngParser.C.

References `isColor()`, `isGray()`, and `luminance()`.

**10.208.3.4   [Image](#)< [PixRGB](#)< [byte](#) > > PngParser::getRGB ()**

Get an RGB image from the file.

This works regardless of the actual mode of the underlying file (i.e., if the file was grayscale, this will convert that to RGB).

Definition at line 217 of file PngParser.C.

References [isColor\(\)](#), [isGray\(\)](#), and [toRGB\(\)](#).

**10.208.3.5   [bool](#) PngParser::isColor () const**

Query whether the image file is color (RGB or color palette).

Definition at line 185 of file PngParser.C.

Referenced by [getFloat\(\)](#), [getGray\(\)](#), and [getRGB\(\)](#).

**10.208.3.6   [bool](#) PngParser::isGray () const**

Query whether the image file is grayscale.

Definition at line 178 of file PngParser.C.

Referenced by [getFloat\(\)](#), [getGray\(\)](#), and [getRGB\(\)](#).

The documentation for this class was generated from the following files:

- [PngParser.H](#)
- [PngParser.C](#)

## 10.209 PngWriter Class Reference

Helper class for PNG-writing functions.

### Public Member Functions

- void **open** (const char \*[filename](#))
- void **close** ()
- void **writeGray** (const [Image](#)< [byte](#) > &img)
- void **writeRGB** (const [Image](#)< [PixRGB](#)< [byte](#) > > &img)

### 10.209.1 Detailed Description

Helper class for PNG-writing functions.

Essentially this is here to make proper error cleanup easier, e.g. in the presence of exceptions.

Definition at line 51 of file PngWriter.C.

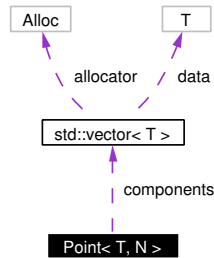
The documentation for this class was generated from the following file:

- [PngWriter.C](#)

## 10.210 **Point< T, N > Class Template Reference**

```
#include <Point.H>
```

Collaboration diagram for Point< T, N >:



### Public Member Functions

- [Point](#) (T filler)  
*Use this to set all the components to a particular value.*
- template<class SourceT> [Point](#) (const SourceT array[N])  
*Use an array to initialize the point.*
- template<class SourceT> **Point** (const [std::vector](#)< SourceT > varray)
- template<class SourceT, int sourceDimensionParam> **Point** (const [Point](#)< SourceT, sourceDimensionParam > &source)
- template<class SourceT> **Point** (const [Point](#)< SourceT, N > &source)
- const [std::vector](#)< T > [getComponents](#) (void) const  
*Returns component array.*
- [std::vector](#)< T > [getComponents](#) (void)
- template<class SourceT> void [setComponents](#) (const [std::vector](#)< SourceT > a)
- T [operator\[ \]](#) (int index) const  
*return the i'th component*
- T & [operator\[ \]](#) (int index)
- T [sqr](#) (void) const
- double [mag](#) (void) const
- T [abs](#) (void) const  
*Returns absolute norm.*

- `T max` (void) const

*Returns the maximum norm of a component array.*

- `Point & operator+=` (const `Point` &p)
- `Point operator+` (const `Point` &p)
- `Point & operator-=` (const `Point` &p)
- `Point operator-` (const `Point` &p)
- `Point operator *` (const `Point` &p)
- `Point & operator *=` (const `Point` &p)
- `Point & operator *=` (const `T` &p)

*dot product*

- `Point operator *` (const `T` &p)
- `Point operator/` (const `Point` &p)

*point-wise division.*

- `Point & operator/=` (const `Point` &p)
- `Point & operator/=` (const `T` &p)
- `Point operator/` (const `T` &p)
- `Point operator=` (const `Point` &p)

## Static Public Attributes

- const int `dimension` = N

## Protected Attributes

- `std::vector< T > components`

### 10.210.1 Detailed Description

`template<class T, int N> class Point< T, N >`

Generic Euclidean Point class T is the type of the point components N is the dimensions of a point.

Definition at line 48 of file Point.H.



## 10.210.2 Constructor & Destructor Documentation

**10.210.2.1** `template<class T, int N> Point< T, N >::Point (T filler)`  
`[inline, explicit]`

Use this to set all the components to a particular value.

Definition at line 60 of file `Point.H`.

**10.210.2.2** `template<class T, int N> template<class SourceT> Point< T, N >::Point (const SourceT array[N])` `[inline]`

Use an array to initialize the point.

Definition at line 69 of file `Point.H`.

## 10.210.3 Member Function Documentation

**10.210.3.1** `template<class T, int N> T Point< T, N >::abs (void) const`  
`[inline]`

Returns absolute norm.

Definition at line 132 of file `Point.H`.

References `abs()`.

**10.210.3.2** `template<class T, int N> const std::vector<T> Point< T, N >::getComponents (void) const` `[inline]`

Returns component array.

Definition at line 88 of file `Point.H`.

**10.210.3.3** `template<class T, int N> T Point< T, N >::max (void) const`  
`[inline]`

Returns the maximum norm of a component array.

Definition at line 141 of file `Point.H`.

References `abs()`.

**10.210.3.4** `template<class T, int N> Point& Point< T, N >::operator*=(const T & p) [inline]`

dot product

Definition at line 212 of file `Point.H`.

**10.210.3.5** `template<class T, int N> Point Point< T, N >::operator/(const Point< T, N > & p) [inline]`

point-wise division.

Definition at line 232 of file `Point.H`.

**10.210.3.6** `]`

`template<class T, int N> T Point< T, N >::operator[] (int index) const [inline]`

return the *i*'th component

Definition at line 105 of file `Point.H`.

The documentation for this class was generated from the following file:

- [Point.H](#)

## 10.211 Point2D Class Reference

This is a basic class to encode 2D integer coordinates.

```
#include <Point2D.H>
```

### Public Member Functions

- [Point2D](#) ()  
*The default constructor initializes the coordinates to (0,0).*
- [Point2D](#) (const int ii, const int jj)  
*Initialize the Point2D from horizontal & vertical coordinates.*
- [Point2D](#) & [operator+=](#) (const [Point2D](#) &p)  
*+= operator*
- [Point2D](#) & [operator-=](#) (const [Point2D](#) &p)  
*-= operator*
- [Point2D](#) & [operator\\*=](#) (const [Point2D](#) &p)  
*\*= operator*
- [Point2D](#) & [operator/=](#) (const [Point2D](#) &p)  
*/= operator*
- [Point2D](#) [operator+](#) (const [Point2D](#) &p) const  
*+ operator*
- [Point2D](#) [operator-](#) (const [Point2D](#) &p) const  
*- operator*
- [Point2D](#) [operator\\*](#) (const [Point2D](#) &p) const  
*\* operator*
- [Point2D](#) [operator/](#) (const [Point2D](#) &p) const  
*/ operator*
- [Point2D](#) & [operator+=](#) (const int val)  
*+= operator*
- [Point2D](#) & [operator-=](#) (const int val)

*-= operator*

- [Point2D](#) & [operator](#) \*= (const int val)  
*\*= operator*
- [Point2D](#) & [operator](#) /= (const int val)  
*/= operator*
- [Point2D](#) [operator](#)+ (const int val) const  
*+ operator*
- [Point2D](#) [operator](#)- (const int val) const  
*- operator*
- [Point2D](#) [operator](#) \* (const int val) const  
*\* operator*
- [Point2D](#) [operator](#)/ (const int val) const  
*/ operator*
- bool [isValid](#) () const  
*test whether i and j are both positive*
- float [squadist](#) (const [Point2D](#) &p) const  
*the square of the euclidean distance*
- float [distance](#) (const [Point2D](#) &p) const  
*the euclidean distance from p*
- [Point2D](#) [fitToDims](#) (const [Dims](#) dims)  
*clamp the coords such that the point fits into image of dimensions dims*
- void [clampToDims](#) (const [Dims](#) dims)  
*clamp the coords such that the point fits into image of dimensions dims*

## Public Attributes

- int [i](#)  
*2D coordinates*
- int [j](#)  
*2D coordinates*

### 10.211.1 Detailed Description

This is a basic class to encode 2D integer coordinates.

This is a completely public class whose goal is just to provide a shorthand notation for 2D integer coordinates. All methods are inlined, so there is no .C file, just a .H file.

Definition at line 49 of file Point2D.H.

### 10.211.2 Constructor & Destructor Documentation

#### 10.211.2.1 Point2D::Point2D () [inline]

The default constructor initializes the coordinates to (0,0).

Definition at line 126 of file Point2D.H.

References `i`, and `j`.

Referenced by `fitToDims()`, `operator*()`, `operator+()`, `operator-()`, and `operator/()`.

#### 10.211.2.2 Point2D::Point2D (const int *ii*, const int *jj*) [inline]

Initialize the Point2D from horizontal & vertical coordinates.

Definition at line 130 of file Point2D.H.

References `i`, and `j`.

### 10.211.3 Member Function Documentation

#### 10.211.3.1 void Point2D::clampToDims (const Dims *dims*) [inline]

clamp the coords such that the point fits into image of dimensions `dims`

fatsre than `fitToDims()` and no return value.

Definition at line 232 of file Point2D.H.

References `Dims::h()`, `i`, `j`, and `Dims::w()`.

Referenced by `SimulationViewerEyeMvt2::eyeMovement()`, `SimulationViewerEyeMvt::eyeMovement()`, and `SimulationViewerCompress::getTraj()`.

#### 10.211.3.2 float Point2D::distance (const Point2D & *p*) const [inline]

the euclidean distance from `p`

Definition at line 217 of file Point2D.H.

References `sqrt()`, and `squdist()`.

Referenced by `SimulationViewerEyeMvt2::eyeMovement()`, `SimulationViewerEyeMvt::eyeMovement()`, `VisualBuffer::findMostInterestingTargetLocMax()`, and `SimulationViewerStd::linkFOAs()`.

#### 10.211.3.3 **Point2D** `Point2D::fitToDims (const Dims dims)` [inline]

clamp the coords such that the point fits into image of dimensions `dims`

returns the difference between the corrected values and the old ones

Definition at line 221 of file `Point2D.H`.

References `Dims::h()`, `i`, `j`, `Point2D()`, and `Dims::w()`.

Referenced by `ShapeEstimator::getObjectMask()`.

#### 10.211.3.4 **bool** `Point2D::isValid ()` const [inline]

test whether `i` and `j` are both positive

Definition at line 206 of file `Point2D.H`.

References `i`, and `j`.

#### 10.211.3.5 **Point2D** `Point2D::operator * (const int val)` const [inline]

`*` operator

Definition at line 190 of file `Point2D.H`.

References `i`, `j`, and `Point2D()`.

#### 10.211.3.6 **Point2D** `Point2D::operator * (const Point2D & p)` const [inline]

`*` operator

Definition at line 158 of file `Point2D.H`.

References `i`, `j`, and `Point2D()`.

#### 10.211.3.7 **Point2D** & `Point2D::operator *= (const int val)` [inline]

`*=` operator

Definition at line 174 of file `Point2D.H`.

References `i`, and `j`.

**10.211.3.8** [Point2D](#) & Point2D::operator \*= (const [Point2D](#) & p) [inline]

\*= operator

Definition at line 142 of file Point2D.H.

References i, and j.

**10.211.3.9** [Point2D](#) Point2D::operator+ (const int val) const [inline]

+ operator

Definition at line 182 of file Point2D.H.

References i, j, and Point2D().

**10.211.3.10** [Point2D](#) Point2D::operator+ (const [Point2D](#) & p) const [inline]

+ operator

Definition at line 150 of file Point2D.H.

References i, j, and Point2D().

**10.211.3.11** [Point2D](#) & Point2D::operator+= (const int val) [inline]

+= operator

Definition at line 166 of file Point2D.H.

References i, and j.

**10.211.3.12** [Point2D](#) & Point2D::operator+= (const [Point2D](#) & p) [inline]

+= operator

Definition at line 134 of file Point2D.H.

References i, and j.

**10.211.3.13** [Point2D](#) Point2D::operator- (const int val) const [inline]

- operator

Definition at line 186 of file Point2D.H.

References i, j, and Point2D().

**10.211.3.14** [Point2D](#) [Point2D::operator-](#) (const [Point2D](#) & *p*) const [inline]

- operator

Definition at line 154 of file Point2D.H.

References [i](#), [j](#), and [Point2D\(\)](#).

**10.211.3.15** [Point2D](#) & [Point2D::operator-=](#) (const int *val*) [inline]

`-=` operator

Definition at line 170 of file Point2D.H.

References [i](#), and [j](#).

**10.211.3.16** [Point2D](#) & [Point2D::operator-=](#) (const [Point2D](#) & *p*) [inline]

`-=` operator

Definition at line 138 of file Point2D.H.

References [i](#), and [j](#).

**10.211.3.17** [Point2D](#) [Point2D::operator/](#) (const int *val*) const [inline]

`/` operator

Definition at line 194 of file Point2D.H.

References [i](#), [j](#), and [Point2D\(\)](#).

**10.211.3.18** [Point2D](#) [Point2D::operator/](#) (const [Point2D](#) & *p*) const [inline]

`/` operator

Definition at line 162 of file Point2D.H.

References [i](#), [j](#), and [Point2D\(\)](#).

**10.211.3.19** [Point2D](#) & [Point2D::operator/=](#) (const int *val*) [inline]

`/=` operator

Definition at line 178 of file Point2D.H.

References [i](#), and [j](#).



**10.211.3.20** `Point2D & Point2D::operator/= (const Point2D & p)` [inline]

`/=` operator

Definition at line 146 of file Point2D.H.

References `i`, and `j`.

**10.211.3.21** `float Point2D::squadist (const Point2D & p) const` [inline]

the square of the euclidean distance

Definition at line 210 of file Point2D.H.

References `i`, and `j`.

Referenced by `ThresholdSaccadeController::checkPercepts()`, `ThresholdSaccadeController::computeWhenNewDecision()`, and `distance()`.

**10.211.4 Member Data Documentation****10.211.4.1** `int Point2D::i`

2D coordinates

Definition at line 112 of file Point2D.H.

Referenced by `VisualBuffer::bufferToRetinal()`, `WTAwinner::buildFromSMcoords()`, `ThresholdSaccadeController::checkPercepts()`, `clampToDims()`, `SurpriseModelSP::combineFrom()`, `SurpriseModelSG::combineFrom()`, `GetSaliency::compute()`, `ModelEyeHeadController::computeWhenNewDecision()`, `ModelWrapperSaccadeController::computeWhenNewDecision()`, `MonkeySaccadeController::computeWhenNewDecision()`, `Monkey2SaccadeController::computeWhenNewDecision()`, `SimpleEyeHeadSaccadeController::computeWhenNewDecision()`, `ThresholdFrictionSaccadeController::computeWhenNewDecision()`, `FrictionSaccadeController::computeWhenNewDecision()`, `ModelWrapperSaccadeController::computeWhenNewDecisionHead()`, `Monkey2SaccadeController::computeWhenNewDecisionHead()`, `SimpleEyeHeadSaccadeController::computeWhenNewDecisionHead()`, `ModelWrapperSaccadeController::computeWhenNewPercept()`, `Monkey2SaccadeController::computeWhenNewPercept()`, `ThresholdFrictionSaccadeController::computeWhenNewPercept()`, `FrictionSaccadeController::computeWhenNewPercept()`, `FrictionSaccadeController::computeWhenResetPos()`, `convertFromString()`, `convertToString()`, `Image< T >::coordsOk()`, `corrpatch()`, `Image< T >::countParticles()`, `SaliencyMapFast::depress()`, `SaliencyMapTrivial::depress()`, `SaliencyMapStd::depress()`, `dilateImg()`, `ImageDisplay::display()`, `PsychoDisplay::displayEyeTrackerCalibration()`, `PsychoDisplay::displayImagePatch()`, `PsychoDisplay::displayText()`, `distDegrade()`, `RetinaStd::doGetOutput()`, `ScorrChannel::doInput()`, `Image< T >::drawArrow()`,

Image< T >::drawCircle(), Image< T >::drawCross(), Image< T >::drawDisk(), Image< T >::drawDiskCheckTarget(), SimulationViewerStd::drawEye(), SimulationViewerStd::drawHead(), Image< T >::drawLine(), SimulationViewerStd::drawMaskOutline(), Image< T >::drawPatch(), TemplateMatchChannel::drawResults(), VisualEventSet::drawTokens(), SaccadeController::dumpQueues(), erodeImg(), WinnerTakeAllDummy::evolve(), WinnerTakeAllFast::evolve(), WinnerTakeAllStd::evolve(), VisualCortexEyeMvt::evolve(), ModelEyeHeadController::evolve(), EyeTrackerSaccadeController::evolve(), MonkeyEyeHeadController::evolve(), FrictionSaccadeController::evolve(), Brain::evolve(), SimulationViewerEyeMvt2::eyeMovement(), SimulationViewerEyeMvt::eyeMovement(), SimulationViewerCompress::eyeMovement(), TemplateMatchChannel::findBestMatch(), Image< T >::findMax(), Image< T >::findMin(), VisualBuffer::findMostInterestingTargetLocMax(), fitToDims(), fixationMask(), Image< T >::flood(), floodCleanBB(), PyrFoveator::foveate(), BlurFoveator::foveate(), gaussianBlob(), ImageSpring< T >::getClusteredImage(), SaccadeController::getDecision(), SaccadeController::getDecisionHead(), MonkeySaccadeController::getEye(), MonkeySaccadeController::getHead(), ImageSpring< T >::getIndex(), BeobotVisualCortex::getInputSize(), MonkeySaccadeController::getJustEye(), ImageSpring< T >::getNeighbor(), RetinaStd::getRawInputRectangle(), getSalRegions(), Image< T >::getSample(), WTAWinner::getSMcoords(), SimulationViewerStd::getTraj(), SimulationViewerEyeMvt2::getTraj(), SimulationViewerCompress::getTraj(), Image< T >::getVal(), BeobotVisualCortex::getWinner(), ImageSpring< T >::getXY(), Beobot::highLevel(), Beobot::intermediateLevel(), WTAWinner::isValid(), Point2DT::isValid(), isValid(), main(), SurpriseImage< T >::neighborhoods(), omniCorrectGen(), operator \*(), operator \*(), operator !=(), operator +(), operator +=(), operator -, operator -=(), operator /(), operator /=(), operator ==(), Image< T >::paste(), Image< T >::pasteGabor(), pasteImage(), ModelEyeHeadController::pix2deg(), MonkeyEyeHeadController::pix2deg(), Point2D(), BeobotVisualCortex::process(), BeobotVisualCortex::processEnd(), RetinaStd::rawToRetinal(), Rectangle::Rectangle(), Vector2D::reset(), VisualBuffer::retinalToBuffer(), RetinaStd::retinalToRaw(), ModelEyeHeadController::saccade(), segmentLandmark(), segmentObject(), segmentObjectClean(), LPTFoveator::setOrigin(), Foveator::setOrigin(), ImageDisplay::setPosition(), Image< T >::setVal(), BeobotVisualCortex::singleCPUprocess(), sqdist(), InputFrameSeries::start2(), TaskRelevanceMapStd::update(), VisualEventSet::updateEvents(), SaliencyMap::w(), and Image< T >::writeText().

#### 10.211.4.2 int Point2D::j

2D coordinates

Definition at line 112 of file Point2D.H.

Referenced by VisualBuffer::bufferToRetinal(), WTAWinner::buildFromSMcoords(), ThresholdSaccadeController::checkPercepts(), clampToDims(), SurpriseModelSP::combineFrom(), SurpriseModelSG::combineFrom(), Model-

WrapperSaccadeController::computeWhenNewDecision(), MonkeySaccadeController::computeWhenNewDecision(), Monkey2SaccadeController::computeWhenNewDecision(), SimpleEyeHeadSaccadeController::computeWhenNewDecision(), FrictionSaccadeController::computeWhenNewDecision(), ModelWrapperSaccadeController::computeWhenNewDecisionHead(), Monkey2SaccadeController::computeWhenNewDecisionHead(), SimpleEyeHeadSaccadeController::computeWhenNewDecisionHead(), ModelWrapperSaccadeController::computeWhenNewPercept(), Monkey2SaccadeController::computeWhenNewPercept(), ThresholdFrictionSaccadeController::computeWhenNewPercept(), FrictionSaccadeController::computeWhenNewPercept(), FrictionSaccadeController::computeWhenResetPos(), convertFromString(), convertToString(), Image< T >::coordsOk(), corrpach(), Image< T >::countParticles(), SaliencyMapFast::depress(), SaliencyMapTrivial::depress(), SaliencyMapStd::depress(), dilateImg(), ImageDisplay::display(), PsychoDisplay::displayEyeTrackerCalibration(), PsychoDisplay::displayImagePatch(), PsychoDisplay::displayText(), distDegrade(), RetinaStd::doGetOutput(), ScorrChannel::doInput(), Image< T >::drawArrow(), Image< T >::drawCircle(), Image< T >::drawCross(), Image< T >::drawDisk(), Image< T >::drawDiskCheckTarget(), SimulationViewerStd::drawEye(), SimulationViewerStd::drawHead(), Image< T >::drawLine(), SimulationViewerStd::drawMaskOutline(), Image< T >::drawPatch(), TemplateMatchChannel::drawResults(), VisualEventSet::drawTokens(), SaccadeController::dumpQueues(), erodeImg(), WinnerTakeAllDummy::evolve(), WinnerTakeAllFast::evolve(), WinnerTakeAllStd::evolve(), VisualCortexEyeMvt::evolve(), ModelEyeHeadController::evolve(), EyeTrackerSaccadeController::evolve(), MonkeyEyeHeadController::evolve(), FrictionSaccadeController::evolve(), Brain::evolve(), SimulationViewerEyeMvt2::eyeMovement(), SimulationViewerEyeMvt::eyeMovement(), SimulationViewerCompress::eyeMovement(), TemplateMatchChannel::findBestMatch(), Image< T >::findMax(), Image< T >::findMin(), VisualBuffer::findMostInterestingTargetLocMax(), fitToDims(), fixationMask(), Image< T >::flood(), floodCleanBB(), PyrFoveator::foveate(), BlurFoveator::foveate(), gaussianBlob(), ImageSpring< T >::getClusteredImage(), SaccadeController::getDecision(), SaccadeController::getDecisionHead(), MonkeySaccadeController::getEye(), MonkeySaccadeController::getHead(), ImageSpring< T >::getIndex(), BeobotVisualCortex::getInputSize(), MonkeySaccadeController::getJustEye(), ImageSpring< T >::getNeighbor(), RetinaStd::getRawInputRectangle(), getSalRegions(), Image< T >::getSample(), WTAWinner::getSMcoords(), SimulationViewerStd::getTraj(), SimulationViewerEyeMvt2::getTraj(), SimulationViewerCompress::getTraj(), Image< T >::getVal(), BeobotVisualCortex::getWinner(), ImageSpring< T >::getXY(), Beobot::highLevel(), Beobot::intermediateLevel(), WTAWinner::isValid(), Point2D::isValid(), isValid(), main(), SurpriseImage< T >::neighborhoods(), omniCorrectGen(), operator \*(), operator \*(), operator !=(), operator +(), operator +=(), operator /(), operator -=(), operator /(), operator ==(), Image< T >::paste(), Image< T >::pasteGabor(), pasteImage(), ModelEyeHeadController::pix2deg(), MonkeyEyeHeadController::pix2deg(), Point2D(), BeobotVisualCortex::process(), BeobotVisualCortex::processEnd(), RetinaStd::rawToRetinal(), Rectangle::Rectangle(), Vector2D::reset(), VisualBuffer::retinalTo

Buffer(), RetinaStd::retinalToRaw(), ModelEyeHeadController::saccade(), segmentLandmark(), segmentObject(), segmentObjectClean(), LPTFoveator::setOrigin(), Foveator::setOrigin(), ImageDisplay::setPosition(), Image< T >::setVal(), BeobotVisualCortex::singleCPUprocess(), squidist(), InputFrameSeries::start2(), TaskRelevanceMapStd::update(), VisualEventSet::updateEvents(), SaliencyMap::w(), and Image< T >::writeText().

The documentation for this class was generated from the following file:

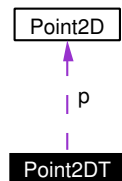
- [Point2D.H](#)

## 10.212 Point2DT Class Reference

This is a fairly trivial (i, j, t) set representing a timestamped [Point2D](#).

```
#include <Point2DT.H>
```

Collaboration diagram for Point2DT:



### Public Member Functions

- [Point2DT](#) ()  
*Default constructor, initializes to zeros.*
- [Point2DT](#) (const [Point2D](#) &pp, const double tt)  
*Constructor given a [Point2D](#) and time.*
- [Point2DT](#) (const int i, const int j, const double tt)  
*Constructor given (i, j) and time.*
- bool [isValid](#) () const  
*returns true if coordinates are not (-1, -1)*

### Public Attributes

- [Point2D](#) p  
*The spatial coordinates.*
- double t  
*The time, in seconds.*

### 10.212.1 Detailed Description

This is a fairly trivial (i, j, t) set representing a timestamped [Point2D](#).

This class is an open (all members public) container for a [Point2D](#) and a time stamp. Like [Point2D](#), this class is fully inlined, so there is no Point2DT.C file.

Definition at line 47 of file Point2DT.H.

### 10.212.2 Constructor & Destructor Documentation

#### 10.212.2.1 `Point2DT::Point2DT ()` `[inline]`

Default constructor, initializes to zeros.

Definition at line 51 of file Point2DT.H.

References `t`.

#### 10.212.2.2 `Point2DT::Point2DT (const Point2D & pp, const double tt)` `[inline]`

Constructor given a [Point2D](#) and time.

Definition at line 54 of file Point2DT.H.

References `t`.

#### 10.212.2.3 `Point2DT::Point2DT (const int i, const int j, const double tt)` `[inline]`

Constructor given (i, j) and time.

Definition at line 57 of file Point2DT.H.

References `t`.

### 10.212.3 Member Function Documentation

#### 10.212.3.1 `bool Point2DT::isValid () const` `[inline]`

returns true if coordinates are not (-1, -1)

Definition at line 71 of file Point2DT.H.

References `Point2D::i`, and `Point2D::j`.

Referenced by `SimulationViewerStd::drawEye()`, `SimulationViewerStd::drawHead()`, `SimulationViewerStd::linkEyes()`, `SimulationViewerStd::linkHeads()`, and `main()`.

## 10.212.4 Member Data Documentation

### 10.212.4.1 [Point2D](#) `Point2DT::p`

The spatial coordinates.

Definition at line 64 of file `Point2DT.H`.

Referenced by `ModelEyeHeadController::computeWhenNewDecision()`, `MonkeySaccadeController::computeWhenNewDecision()`, `MonkeyEyeHeadController::computeWhenNewDecision()`, `ThresholdFrictionSaccadeController::computeWhenNewDecision()`, `ThresholdSaccadeController::computeWhenNewDecision()`, `SimulationViewerStd::drawEye()`, `SimulationViewerStd::drawHead()`, `SaccadeController::dumpQueues()`, `SimulationViewerEyeMvt2::eyeMovement()`, `SimulationViewerEyeMvt::eyeMovement()`, `SimulationViewerCompress::eyeMovement()`, `MonkeySaccadeController::getEye()`, `MonkeySaccadeController::getHead()`, `MonkeySaccadeController::getJustEye()`, `SimulationViewerStd::linkEyes()`, `SimulationViewerStd::linkHeads()`, `main()`, and `SaccadeController::resetPos()`.

### 10.212.4.2 `double` [Point2DT::t](#)

The time, in seconds.

Definition at line 65 of file `Point2DT.H`.

Referenced by `ThresholdFrictionSaccadeController::computeWhenNewDecision()`, `SaccadeController::dumpQueues()`, `MonkeySaccadeController::getEye()`, `MonkeySaccadeController::getHead()`, `MonkeySaccadeController::getJustEye()`, `Point2DT()`, and `SaccadeController::resetPos()`.

The documentation for this class was generated from the following file:

- [Point2DT.H](#)

## 10.213 `promote_trait< T1, T2 >` Struct Template Reference

Promote from T1 and T2 to a type than can hold  $T1 * T2$ .

```
#include <Promotions.H>
```

### Public Types

- `typedef T1 TP`

#### 10.213.1 Detailed Description

```
template<class T1, class T2> struct promote_trait< T1, T2 >
```

Promote from T1 and T2 to a type than can hold  $T1 * T2$ .

Basic promotion mechanism: given T1 and T2, TP provides the appropriate type to hold the result of an operation involving T1 and T2. Default is to promote to type T1, that is, no change (this is used when T2 is "weaker" than T1; when T2 is "stronger" than T1, the explicit specialized rules below are used).

Definition at line 66 of file Promotions.H.

The documentation for this struct was generated from the following file:

- [Promotions.H](#)

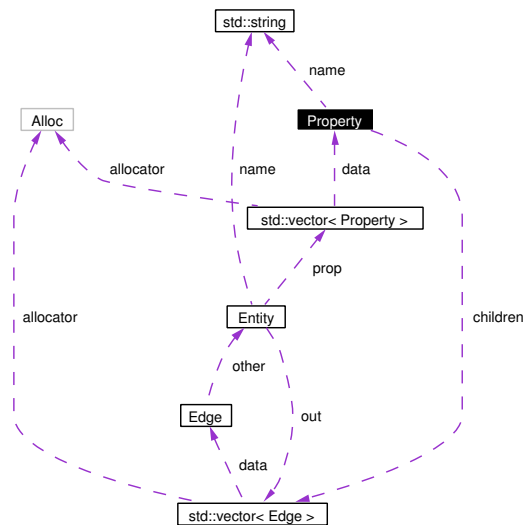


## 10.214 Property Class Reference

Definition of Property class and its functions.

```
#include <Graph.H>
```

Collaboration diagram for Property:



### Public Member Functions

- [Property \(\)](#)  
*Constructor.*
- [~Property \(\)](#)  
*Destructor.*
- void [set\\_name \(std::string name\)](#)
- [std::string get\\_name \(\)](#) const  
*get name of property*
- void [add\\_child \(const Edge &edge\)](#)
- const [std::vector< Edge > get\\_children \(\)](#) const  
*get the list of children*

### 10.214.1 Detailed Description

Definition of Property class and its functions.

This class contains the set of edges that lead to all entities that share a particular property. Currently, properties are restricted to "Big", "Small", "Natural", "Artificial".

Definition at line 67 of file Graph.H.

### 10.214.2 Constructor & Destructor Documentation

#### 10.214.2.1 Property::Property () [inline]

Constructor.

Definition at line 260 of file Graph.H.

#### 10.214.2.2 Property::~Property () [inline]

Destructor.

Definition at line 264 of file Graph.H.

### 10.214.3 Member Function Documentation

#### 10.214.3.1 void Property::add\_child (const Edge & edge) [inline]

add a child to list of children

**Parameters:**

*edge* enter the edge leading to child entity

Definition at line 276 of file Graph.H.

#### 10.214.3.2 const std::vector< Edge > Property::get\_children () const [inline]

get the list of children

Definition at line 280 of file Graph.H.

#### 10.214.3.3 std::string Property::get\_name () const [inline]

get name of property

Definition at line 272 of file Graph.H.

#### 10.214.3.4 void Property::set\_name (std::string name) [inline]

set name of property

**Parameters:**

*name* enter name of property

Definition at line 268 of file Graph.H.

The documentation for this class was generated from the following file:

- [Graph.H](#)

## 10.215 PropertyVectorSet Class Reference

class that contains a set of property vectors for events

```
#include <VisualEvent.H>
```

### Public Member Functions

- [PropertyVectorSet \(\)](#)  
*default constructor*
- [PropertyVectorSet \(std::istream &is\)](#)  
*read from stream constructor*
- void [writeToStream](#) (std::ostream &os) const  
*write the entire PropertyVectorSet to the output stream os*
- void [readFromStream](#) (std::istream &is)  
*read the PropertyVectorSet from the input stream is*
- [std::vector< float > getPropertyVectorForEvent](#) (const int num)  
*return the PropertyVector for an event by its number*

### Public Attributes

- [std::vector< std::vector< float > > itsVectors](#)  
*the vector of property vectors*

#### 10.215.1 Detailed Description

class that contains a set of property vectors for events

each property vector contains the following fields:

| index | content               |
|-------|-----------------------|
| 0     | event number          |
| 1     | interestingness index |
| 2     | area                  |
| 3     | Uxx                   |
| 4     | Uyy                   |
| 5     | Uxy                   |
| 6     | major Axis            |
| 7     | minor Axis            |
| 8     | elongation            |
| 9     | orientation angle     |
| 10    | maximum intensity     |
| 11    | minimum intensity     |
| 12    | average intensity     |

Definition at line 120 of file VisualEvent.H.

## 10.215.2 Constructor & Destructor Documentation

### 10.215.2.1 PropertyVectorSet::PropertyVectorSet ()

default constructor

Definition at line 121 of file VisualEvent.C.

### 10.215.2.2 PropertyVectorSet::PropertyVectorSet (std::istream & is)

read from stream constructor

Definition at line 125 of file VisualEvent.C.

References readFromStream().

## 10.215.3 Member Function Documentation

### 10.215.3.1 `std::vector< float > PropertyVectorSet::getPropertyVectorForEvent (const int num)`

return the PropertyVector for an event by its number

Definition at line 161 of file VisualEvent.C.

References itsVectors, and uint.

Referenced by VisualEventSet::drawTokens().

**10.215.3.2 void PropertyVectorSet::readFromStream (std::istream & is)**

read the PropertyVectorSet from the input stream is

Definition at line 149 of file VisualEvent.C.

References itsVectors, and uint.

Referenced by MbariResultViewer::loadProperties(), and PropertyVectorSet().

**10.215.3.3 void PropertyVectorSet::writeToStream (std::ostream & os) const**

write the entire PropertyVectorSet to the output stream os

Definition at line 131 of file VisualEvent.C.

References itsVectors, and uint.

Referenced by MbariResultViewer::saveProperties().

**10.215.4 Member Data Documentation****10.215.4.1 [std::vector< \[std::vector<float>\]\(#\) >](#) [PropertyVectorSet::itsVectors](#)**

the vector of property vectors

Definition at line 140 of file VisualEvent.H.

Referenced by getPropertyVectorForEvent(), VisualEventSet::getPropertyVectorSet(), readFromStream(), and writeToStream().

The documentation for this class was generated from the following files:

- [VisualEvent.H](#)
- [VisualEvent.C](#)

## 10.216 PropHold Class Reference

this is a container class for the property at each hyper pixel for [ContourNeuronCreate](#)

```
#include <contourNeuron.H>
```

### Public Attributes

- float [mis](#)  
*misalignement modifier*
- float [ang](#)  
*Beta angle modifier.*
- float [ang2](#)  
*Alpha angle modifier.*
- float [ang3](#)  
*Aux angle modifier.*
- float [angAB](#)  
*Product of Alpha and Beta.*
- float [angABD](#)  
*Product of Alpha,Beta and distance.*
- float [dis](#)  
*Generic distance modifier.*
- float [Edis](#)  
*Excitation distance modifier.*
- float [Sdis](#)  
*Supression distance modifier.*
- float [Odis](#)  
*orthogonal distance modifier*
- bool [pol](#)  
*polarity of hyper pixel*
- bool [sender](#)

*True if hyper pixel can pass on energy.*

- bool [coLinear](#)

*true if this pixel is co-linear to the perfered angle*

- bool [fullOrth](#)

*true if this pixel is fully orthogonoal to the perfered angle*

- bool [zero](#)

*True if this hyper pixel is non-zero.*

- bool [orth](#)

*This is true if the element is and orthogonal excitation connection.*

### 10.216.1 Detailed Description

this is a container class for the property at each hyper pixel for [ContourNeuronCreate](#)

This class is used to hold the properties of each hyper pixel in in the 4D neural template. each of the floats used here are essentilly weights used when a pixel in the [ImageMap](#) is run against each of the other pixels. In essence it simulates the connection weights of each neuron to each other neuron. This is used by [ContourNeuronCreate](#).

Definition at line 70 of file contourNeuron.H.

### 10.216.2 Member Data Documentation

#### 10.216.2.1 float [PropHold::ang](#)

Beta angle modifier.

Definition at line 76 of file contourNeuron.H.

Referenced by `ContourNeuronCreate< TC2 >::CreateNeuron()`, and `main()`.

#### 10.216.2.2 float [PropHold::ang2](#)

Alpha angle modifier.

Definition at line 78 of file contourNeuron.H.

Referenced by `ContourNeuronCreate< TC2 >::CreateNeuron()`, and `main()`.



**10.216.2.3 float [PropHold::ang3](#)**

Aux angle modifier.

Definition at line 80 of file contourNeuron.H.

Referenced by ContourNeuronCreate< TC2 >::CreateNeuron().

**10.216.2.4 float [PropHold::angAB](#)**

Product of Alpha and Beta.

Definition at line 82 of file contourNeuron.H.

Referenced by ContourNeuronCreate< TC2 >::CreateNeuron(), and main().

**10.216.2.5 float [PropHold::angABD](#)**

Product of Alpha,Beta and distance.

Definition at line 84 of file contourNeuron.H.

Referenced by CINNIC::convolveTest(), ContourNeuronCreate< TC2 >::CreateNeuron(), main(), and CINNIC::viewNeuronTemplate().

**10.216.2.6 bool [PropHold::coLinear](#)**

true if this pixel is co-linear to the perferred angle

Definition at line 100 of file contourNeuron.H.

Referenced by ContourNeuronCreate< TC2 >::CreateNeuron(), and main().

**10.216.2.7 float [PropHold::dis](#)**

Generic distance modifier.

Definition at line 86 of file contourNeuron.H.

Referenced by ContourNeuronCreate< TC2 >::CreateNeuron(), and main().

**10.216.2.8 float [PropHold::Edis](#)**

Excitation distance modifier.

Definition at line 88 of file contourNeuron.H.

Referenced by ContourNeuronCreate< TC2 >::CreateNeuron().

**10.216.2.9    bool [PropHold::fullOrth](#)**

true if this pixel is fully orthogonoal to the perfered angle

Definition at line 102 of file contourNeuron.H.

**10.216.2.10    float [PropHold::mis](#)**

misalignement modifier

Definition at line 74 of file contourNeuron.H.

**10.216.2.11    float [PropHold::Odis](#)**

orthogonal distance modifier

Definition at line 92 of file contourNeuron.H.

Referenced by ContourNeuronCreate< TC2 >::CreateNeuron().

**10.216.2.12    bool [PropHold::orth](#)**

This is true if the element is and orthogonal excitation connection.

Definition at line 107 of file contourNeuron.H.

Referenced by ContourNeuronCreate< TC2 >::CreateNeuron().

**10.216.2.13    bool [PropHold::pol](#)**

polarity of hyper pixel

Definition at line 96 of file contourNeuron.H.

Referenced by ContourNeuronCreate< TC2 >::CreateNeuron(), and main().

**10.216.2.14    float [PropHold::Sdis](#)**

Supression distance modifier.

Definition at line 90 of file contourNeuron.H.

Referenced by ContourNeuronCreate< TC2 >::CreateNeuron().

**10.216.2.15    bool [PropHold::sender](#)**

True if hyper pixel can pass on energy.

Definition at line 98 of file contourNeuron.H.

Referenced by ContourNeuronCreate< TC2 >::CreateNeuron(), and main().

#### 10.216.2.16 bool [PropHold::zero](#)

True if this hyper pixel is non-zero.

Definition at line 104 of file contourNeuron.H.

Referenced by ContourNeuronCreate< TC2 >::CreateNeuron().

The documentation for this class was generated from the following file:

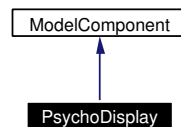
- [contourNeuron.H](#)

## 10.217 PsychoDisplay Class Reference

Class fo do various psychophysics displays.

```
#include <PsychoDisplay.H>
```

Inheritance diagram for PsychoDisplay:



Collaboration diagram for PsychoDisplay:

### Public Member Functions

#### Constructors, destructor and initialization

- `PsychoDisplay (ModelManager &mgr, const std::string &descrName="Psycho Display", const std::string &tagName="PsychoDisplay")`  
*Constructor.*
- `~PsychoDisplay ()`  
*Destructor.*
- `virtual void exportOptions (const int whatToExport, const bool recurse=true)`  
*Export some of our [ModelParam](#) parameters as command-line-options.*

#### Display and interaction functions

- `void clearScreen (const bool vsync=true)`  
*clear the screen*
- `int waitForKey ()`  
*wait for a keypress and return character for key pressed*
- `int checkForKey ()`  
*check whether a keypress has occurred and return char for key pressed*
- `void displayFixation (const int x=-1, const int y=-1, const bool vsync=true)`  
*display a fixation cross*

- void [displayFixationBlink](#) (const int x=-1, const int y=-1, const int iter=5, const int delay=2)  
*blink a fixation cross*
- void [displayISCANcalib](#) ()  
*display a static low-level 5-point ISCAN calibration array*
- void [displayEyeTrackerCalibration](#) (const int nptshoriz=7, const int nptsvertic=5)  
*display a bunch of eye-tracker calibration dots*
- SDL\_Surface \* [makeBlittableSurface](#) (const [Image](#)< [PixRGB](#)< byte > > &img, const bool resiz=false)  
*Make a blittable surface from an [Image](#).*
- void [displayImage](#) (SDL\_Surface \*img, const int frame=-1, const bool vsync=true)  
*display an image*
- void [displayImagePatch](#) (const [Image](#)< [PixRGB](#)< byte > > &image, const [Point2D](#) &pos, const int frame=-1, const bool vsync=true, const bool flip=true)  
*display an image patch*
- void [displayText](#) (const [std::string](#) &msg, const bool vsync=true)  
*show a text message*
- void [waitVsync](#) (const bool checkdelay=false, const bool quiet=true)  
*Wait until start of vertical blanking.*
- void [waitFrames](#) (const int n)  
*Wait for a number of frame periods.*

### Eye tracking functions

- void [trackEyes](#) (const bool startstop)  
*start/stop eye tracking*

### Event logging functions

- void [pushEvent](#) (const [std::string](#) &msg)  
*push an event into our queue*
- void [pushEventBegin](#) (const [std::string](#) &msg)

*push the beginning of an event into our queue*

- void [pushEventEnd](#) (const [std::string](#) &msg)  
*push the end of an event into our queue*

### YUV overlay functions

- void [createYUVoverlay](#) ()  
*create an YUV overlay*
- void [destroyYUVoverlay](#) ()  
*destroy a previously-created YUV overlay*
- [SDL\\_Overlay](#) \* [lockYUVoverlay](#) ()  
*Lock YUV overlay and get a pointer to it for direct access to pixel data.*
- void [unlockYUVoverlay](#) ()  
*Unlock YUV overlay.*
- void [displayYUVoverlay](#) (const int frame, const bool vsync)  
*display current YUV overlay*

### Access functions

- [Dims](#) [getDims](#) () const  
*Get display dimensions.*
- int [getGrey](#) () const  
*Get background grey value.*

### Protected Member Functions

- void [start2](#) ()  
*get started*
- void [stop1](#) ()  
*get stopped*

## Protected Attributes

- `ModelParam< Dims > itsDims`  
*screen resolution*
- `ModelParam< int > itsPriority`  
*priority for SCHED\_FIFO, or 0 for normal*
- `ModelParam< int > itsGrey`  
*greylevel for background grey*
- `ModelParam< int > itsBlack`  
*greylevel for black*
- `ModelParam< int > itsWhite`  
*greylevel for white*
- `ModelParam< std::string > itsFileName`  
*filename for time data*
- `ModelParam< float > itsRefreshDelay`  
*desired refresh delay in usec*
- `ModelParam< int > itsFixSiz`  
*size of fixation cross*
- `ModelParam< int > itsFixThick`  
*thickness of fixation cross*

### 10.217.1 Detailed Description

Class fo do various psychophysics displays.

This class is to facilitate the display of various psychophysics stimuli, with an initial emphasis on playing movies at a controlled framerate. Programs using this class, such as `psycho-movie.C`, should run as root if `SCHED_FIFO` scheduling is required (and I highly recommend it, as it will make timing reliable). The class uses the SDL library to do the displays. This class attempts to lock onto the display's vertical blanking for reliable timing of movie playing. Note that, as far as I know, SDL does not provide a way to select a screen refresh rate; instead, you need to specify the refresh rate in your X config. I used the modeline calculator at <http://xtiming.sourceforge.net/cgi-bin/xtiming.pl> to figure out

good modelines for the refresh rates I wanted. For example: for a 120Hz 640x480 display, I got:

Modeline "640x480@120" 55.43 640 672 880 912 480 487 497 505

which I added to the "Monitor" section of /etc/X11/XF86Config-4, and I also specified that this mode should be used by giving its name in the "Screen" section:

Subsection "Display" Depth 24 Modes "1280x1024" "1280x960" "1152x864" "1024x768" "800x600" "640x480@120" EndSubsection

Always verify on the OSD info page of your monitor that you are actually getting the refresh rate you want.

For 640x480 @ 60Hz doublescan, I use: Modeline "640x480@60d" 48.21 640 672 760 792 480 490 495 505 doublescan

For 800x600 @ 60Hz (for LCD goggles): ModeLine "800x600@60" 40.0 800 840 968 1056 600 601 605 628 +hsync +vsync

Definition at line 85 of file PsychoDisplay.H.

## 10.217.2 Constructor & Destructor Documentation

**10.217.2.1 PsychoDisplay::PsychoDisplay ([ModelManager](#) & mgr, const [std::string](#) & descrName = "Psycho Display", const [std::string](#) & tagName = "PsychoDisplay")**

Constructor.

Definition at line 56 of file PsychoDisplay.C.

References [ModelComponent::addSubComponent\(\)](#).

**10.217.2.2 PsychoDisplay::~PsychoDisplay ()**

Destructor.

Definition at line 81 of file PsychoDisplay.C.

## 10.217.3 Member Function Documentation

**10.217.3.1 int PsychoDisplay::checkForKey ()**

check whether a keypress has occurred and return char for key pressed

a value of -1 is returned if no keypress occurred

Definition at line 259 of file PsychoDisplay.C.



References `pushEvent()`.

### 10.217.3.2 void PsychoDisplay::clearScreen (const bool vsync = true)

clear the screen

**Parameters:**

**vsync** will attempt to sync with vertical blanking if true

Definition at line 225 of file `PsychoDisplay.C`.

References `ModelParam< int >::getVal()`, `ModelParam< Dims >::getVal()`, `Dims::h()`, `itsGrey`, `pushEventBegin()`, `pushEventEnd()`, and `Dims::w()`.

Referenced by `displayEyeTrackerCalibration()`, `displayFixationBlink()`, and `start2()`.

### 10.217.3.3 void PsychoDisplay::createYUVoverlay ()

create an YUV overlay

All the other overlay functions will throw a fatal error if no overlay has been created.

Definition at line 206 of file `PsychoDisplay.C`.

References `destroyYUVoverlay()`, `ModelParam< Dims >::getVal()`, `Dims::h()`, and `Dims::w()`.

### 10.217.3.4 void PsychoDisplay::destroyYUVoverlay ()

destroy a previously-created YUV overlay

Overlay displays and normal displays don't mix well in SDL. So, typically, you would create the overlay just before playing a movie in YUV mode, then play the movie, then clear the screen and destroy the overlay, do a few normal displays (like a fixation cross, some text, etc), and loop.

Definition at line 217 of file `PsychoDisplay.C`.

Referenced by `createYUVoverlay()`, and `stop1()`.

### 10.217.3.5 void PsychoDisplay::displayEyeTrackerCalibration (const int npts<sub>horiz</sub> = 7, const int npts<sub>vertic</sub> = 5)

display a bunch of eye-tracker calibration dots

Definition at line 373 of file `PsychoDisplay.C`.

References `clearScreen()`, `displayFixation()`, `displayFixationBlink()`, `ModelParam< int >::getVal()`, `ModelParam< Dims >::getVal()`, `Dims::h()`, `Point2D::i`, `itsBlack`, `itsWhite`, `Point2D::j`, `pushEventBegin()`, `pushEventEnd()`, `trackEyes()`, `uint`, `Dims::w()`, `waitForKey()`, and `waitVsync()`.

#### 10.217.3.6 **void PsychoDisplay::displayFixation (const int x = -1, const int y = -1, const bool vsync = true)**

display a fixation cross

##### Parameters:

- x** horizontal position of center of cross, or -1 to have it at the center of the screen
- y** vertical position of center of cross, or -1 to have it at the center of the screen
- vsync** will attempt to sync with vertical blanking if true

Definition at line 280 of file `PsychoDisplay.C`.

References `ModelParam< Dims >::getVal()`, `ModelParam< int >::getVal()`, `Dims::h()`, `itsBlack`, `itsFixSiz`, `itsFixThick`, `pushEventBegin()`, `pushEventEnd()`, and `Dims::w()`.

Referenced by `displayEyeTrackerCalibration()`, and `displayFixationBlink()`.

#### 10.217.3.7 **void PsychoDisplay::displayFixationBlink (const int x = -1, const int y = -1, const int iter = 5, const int delay = 2)**

blink a fixation cross

##### Parameters:

- iter** number of times we should blink the cross
- delay** blink duration, in frames as per our frame duration [ModelParam](#)

Definition at line 321 of file `PsychoDisplay.C`.

References `clearScreen()`, `displayFixation()`, `pushEventBegin()`, `pushEventEnd()`, and `waitVsync()`.

Referenced by `displayEyeTrackerCalibration()`.

#### 10.217.3.8 **void PsychoDisplay::displayImage (SDL\_Surface \* img, const int frame = -1, const bool vsync = true)**

display an image

**Parameters:**

**img** a pre-formatted image; no rescaling will be attempted and it will just be blitted into the screen. Suitable surfaces may be obtained by [makeBlittableSurface\(\)](#)

**frame** this is just used for the log messages; if it is -1 then no frame number/duration are logged; if it is -2 both the start and end of the function execution are logged

**vsync** will attempt to sync with vertical blanking if true

Definition at line 489 of file PsychoDisplay.C.

References [pushEvent\(\)](#), [pushEventBegin\(\)](#), [pushEventEnd\(\)](#), and [waitVsync\(\)](#).

Referenced by [displayText\(\)](#).

**10.217.3.9 void PsychoDisplay::displayImagePatch (const [Image](#)< [PixRGB](#)< [byte](#) > > & image, const [Point2D](#) & pos, const int frame = -1, const bool vsync = true, const bool flip = true)**

display an image patch

**Parameters:**

**image** an image patch

**pos** the position of the top-left corner of the image

**frame** this is just used for the log messages; if it is -1 then no frame number/duration are logged; if it is -2 both the start and end of the function execution are logged

**vsync** will attempt to sync with vertical blanking if true

Definition at line 513 of file PsychoDisplay.C.

References [Image](#)< [PixRGB](#)< float > >::getHeight(), [Image](#)< [PixRGB](#)< float > >::getWidth(), [Point2D](#)::i, [image](#), [Point2D](#)::j, [pushEvent\(\)](#), [pushEventBegin\(\)](#), [pushEventEnd\(\)](#), and [waitVsync\(\)](#).

**10.217.3.10 void PsychoDisplay::displayISCANcalib ()**

display a static low-level 5-point ISCAN calibration array

Definition at line 336 of file PsychoDisplay.C.

References [ModelParam](#)< int >::getVal(), [ModelParam](#)< [Dims](#) >::getVal(), [Dims](#)::h(), [itsBlack](#), [itsWhite](#), [pushEventBegin\(\)](#), [pushEventEnd\(\)](#), and [Dims](#)::w().

**10.217.3.11 void PsychoDisplay::displayText (const [std::string](#) & msg, const bool vsync = true)**

show a text message

Definition at line 590 of file PsychoDisplay.C.

References [Image< T >::clear\(\)](#), [displayImage\(\)](#), [Image< T >::getArrayPtr\(\)](#), [Image< T >::getHeight\(\)](#), [ModelParam< int >::getVal\(\)](#), [ModelParam< Dims >::getVal\(\)](#), [Image< T >::getWidth\(\)](#), [Dims::h\(\)](#), [Point2D::i](#), [itsBlack](#), [itsGrey](#), [Point2D::j](#), [makeRGB\(\)](#), [NO\\_INIT](#), [Dims::w\(\)](#), and [Image< T >::writeText\(\)](#).

**10.217.3.12 void PsychoDisplay::displayYUVoverlay (const int frame, const bool vsync)**

display current YUV overlay

Definition at line 565 of file PsychoDisplay.C.

References [ModelParam< Dims >::getVal\(\)](#), [Dims::h\(\)](#), [pushEvent\(\)](#), [Dims::w\(\)](#), and [waitVsync\(\)](#).

**10.217.3.13 void PsychoDisplay::exportOptions (const int whatToExport, const bool recurse = true) [virtual]**

Export some of our [ModelParam](#) parameters as command-line-options.

See the base function in [ModelComponent.H](#) for info.

Reimplemented from [ModelComponent](#).

Definition at line 85 of file PsychoDisplay.C.

References [ModelComponent::exportOptions\(\)](#), [itsFileName](#), [itsPriority](#), [itsRefreshDelay](#), [OPTEXP\\_CORE](#), and [ModelManager::requestOption\(\)](#).

**10.217.3.14 [Dims](#) PsychoDisplay::getDims () const**

Get display dimensions.

Definition at line 732 of file PsychoDisplay.C.

References [ModelParam< Dims >::getVal\(\)](#).

**10.217.3.15 int PsychoDisplay::getGrey () const**

Get background grey value.

Definition at line 736 of file PsychoDisplay.C.

References `ModelParam< int >::getVal()`, and `itsGrey`.

### 10.217.3.16 `SDL_Overlay * PsychoDisplay::lockYUVoverlay ()`

Lock YUV overlay and get a pointer to it for direct access to pixel data.

And what can you do with the pixel data? see `psycho-movie.C` and `pvisionTCP3-master.C` for examples of how to fill it from either raw MPEG movie frames or from an [Image<PixRGB<byte> >](#).

Definition at line 550 of file `PsychoDisplay.C`.

### 10.217.3.17 `SDL_Surface * PsychoDisplay::makeBlittableSurface (const Image< PixRGB< byte > > & img, const bool resize = false)`

Make a blittable surface from an [Image](#).

The resulting surface can be displayed using `displayImage`. It should be freed using `SDL_FreeSurface()`

#### Parameters:

***img*** the image to be displayed

***resize*** if true, the image will be resized such as to fill-up at least one dimension of the display (but aspect ratio will be maintained, and our grey value will be used to fill possible borders)

Definition at line 450 of file `PsychoDisplay.C`.

References `Image< PixRGB< float > >::clear()`, `Image< PixRGB< float > >::embed()`, `Image< PixRGB< float > >::getArrayPtr()`, `Image< PixRGB< float > >::getHeight()`, `ModelParam< int >::getVal()`, `ModelParam< Dims >::getVal()`, `Image< PixRGB< float > >::getWidth()`, `Dims::h()`, `image`, `itsGrey`, `NO_INIT`, `Image< PixRGB< float > >::paste()`, and `Dims::w()`.

### 10.217.3.18 `void PsychoDisplay::pushEvent (const std::string & msg)`

push an event into our queue

Definition at line 644 of file `PsychoDisplay.C`.

References `Timer::get()`.

Referenced by `checkForKey()`, `displayImage()`, `displayImagePatch()`, `displayYUVoverlay()`, `pushEventBegin()`, `pushEventEnd()`, `start2()`, and `trackEyes()`.

**10.217.3.19 void PsychoDisplay::pushEventBegin (const [std::string](#) & msg)**

push the beginning of an event into our queue

This is like [pushEvent\(\)](#) but it will add a start marker to the string

Definition at line 652 of file PsychoDisplay.C.

References [pushEvent\(\)](#).

Referenced by [clearScreen\(\)](#), [displayEyeTrackerCalibration\(\)](#), [displayFixation\(\)](#), [displayFixationBlink\(\)](#), [displayImage\(\)](#), [displayImagePatch\(\)](#), [displayISCANcalib\(\)](#), [waitForKey\(\)](#), and [waitVsync\(\)](#).

**10.217.3.20 void PsychoDisplay::pushEventEnd (const [std::string](#) & msg)**

push the end of an event into our queue

This is like [pushEvent\(\)](#) but it will add an end marker to the string

Definition at line 656 of file PsychoDisplay.C.

References [pushEvent\(\)](#).

Referenced by [clearScreen\(\)](#), [displayEyeTrackerCalibration\(\)](#), [displayFixation\(\)](#), [displayFixationBlink\(\)](#), [displayImage\(\)](#), [displayImagePatch\(\)](#), [displayISCANcalib\(\)](#), [waitForKey\(\)](#), and [waitVsync\(\)](#).

**10.217.3.21 void PsychoDisplay::start2 () [protected, virtual]**

get started

Reimplemented from [ModelComponent](#).

Definition at line 100 of file PsychoDisplay.C.

References [clearScreen\(\)](#), [ModelParam< Dims >::getVal\(\)](#), [ModelParam< int >::getVal\(\)](#), [Dims::h\(\)](#), [itsPriority](#), [pushEvent\(\)](#), [Timer::reset\(\)](#), and [Dims::w\(\)](#).

**10.217.3.22 void PsychoDisplay::stop1 () [protected, virtual]**

get stopped

Reimplemented from [ModelComponent](#).

Definition at line 157 of file PsychoDisplay.C.

References [destroyYUVoverlay\(\)](#), [ModelParam< int >::getVal\(\)](#), [itsFileName](#), and [itsPriority](#).

**10.217.3.23 void PsychoDisplay::trackEyes (const bool *startstop*)**

start/stop eye tracking

Definition at line 628 of file PsychoDisplay.C.

References `pushEvent()`.

Referenced by `displayEyeTrackerCalibration()`.

**10.217.3.24 void PsychoDisplay::unlockYUVoverlay ()**

Unlock YUV overlay.

Definition at line 558 of file PsychoDisplay.C.

**10.217.3.25 int PsychoDisplay::waitForKey ()**

wait for a keypress and return character for key pressed

Definition at line 239 of file PsychoDisplay.C.

References `pushEventBegin()`, and `pushEventEnd()`.

Referenced by `displayEyeTrackerCalibration()`.

**10.217.3.26 void PsychoDisplay::waitFrames (const int *n*)**

Wait for a number of frame periods.

This just repeatedly calles [waitVsync\(\)](#)

Definition at line 726 of file PsychoDisplay.C.

References `waitVsync()`.

**10.217.3.27 void PsychoDisplay::waitVsync (const bool *checkdelay* = false, const bool *quiet* = true)**

Wait until start of vertical blanking.

The vertical blanking period is when the monitor's electron gun travels from the bottom of the screen back to the top. This function updates our `itsLastSync` data member with current time at start of vertical blanking pulse. Most display functions have a '`vsync`' argument and will call this function to perform the sync, so you usually never have to call this directly. It is provided here just in case. If `checkdelay` is true, we will log a warning if it has been more than one `itsRefreshDelay` period since the last time we were called. No logging is made if `quiet` is true, otherwise start and end times are logged.

Definition at line 667 of file PsychoDisplay.C.

References `Timer::get()`, `ModelParam<float>::getVal()`, `itsRefreshDelay`, `pushEventBegin()`, `pushEventEnd()`, and `uint64`.

Referenced by `displayEyeTrackerCalibration()`, `displayFixationBlink()`, `displayImage()`, `displayImagePatch()`, `displayYUVoverlay()`, and `waitFrames()`.

## 10.217.4 Member Data Documentation

### 10.217.4.1 `ModelParam<int> PsychoDisplay::itsBlack` [protected]

greylevel for black

Definition at line 272 of file PsychoDisplay.H.

Referenced by `displayEyeTrackerCalibration()`, `displayFixation()`, `displayISCANcalib()`, and `displayText()`.

### 10.217.4.2 `ModelParam<Dims> PsychoDisplay::itsDims` [protected]

screen resolution

Definition at line 269 of file PsychoDisplay.H.

### 10.217.4.3 `ModelParam<std::string> PsychoDisplay::itsFileName` [protected]

filename for time data

Definition at line 274 of file PsychoDisplay.H.

Referenced by `exportOptions()`, and `stop1()`.

### 10.217.4.4 `ModelParam<int> PsychoDisplay::itsFixSiz` [protected]

size of fixation cross

Definition at line 276 of file PsychoDisplay.H.

Referenced by `displayFixation()`.

### 10.217.4.5 `ModelParam<int> PsychoDisplay::itsFixThick` [protected]

thickness of fixation cross

Definition at line 277 of file PsychoDisplay.H.



Referenced by `displayFixation()`.

#### 10.217.4.6 **ModelParam**<int> **PsychoDisplay::itsGrey** [protected]

greylevel for background grey

Definition at line 271 of file `PsychoDisplay.H`.

Referenced by `clearScreen()`, `displayText()`, `getGrey()`, and `makeBlittableSurface()`.

#### 10.217.4.7 **ModelParam**<int> **PsychoDisplay::itsPriority** [protected]

priority for `SCHED_FIFO`, or 0 for normal

Definition at line 270 of file `PsychoDisplay.H`.

Referenced by `exportOptions()`, `start2()`, and `stop1()`.

#### 10.217.4.8 **ModelParam**<float> **PsychoDisplay::itsRefreshDelay** [protected]

desired refresh delay in usec

Definition at line 275 of file `PsychoDisplay.H`.

Referenced by `exportOptions()`, and `waitVsync()`.

#### 10.217.4.9 **ModelParam**<int> **PsychoDisplay::itsWhite** [protected]

greylevel for white

Definition at line 273 of file `PsychoDisplay.H`.

Referenced by `displayEyeTrackerCalibration()`, and `displayISCANcalib()`.

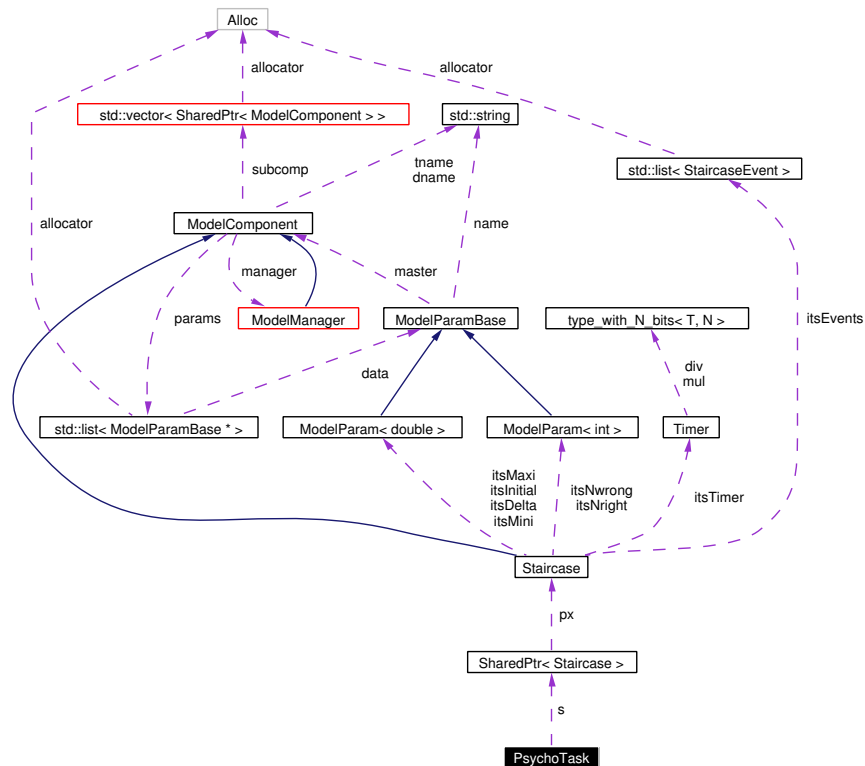
The documentation for this class was generated from the following files:

- [PsychoDisplay.H](#)
- [PsychoDisplay.C](#)

## 10.218 PsychoTask Struct Reference

A summary struct to handle task settings.

Collaboration diagram for PsychoTask:



### Public Attributes

- [SharedPtr< Staircase >](#) s
- [TaskType](#) tt
- double theta
- double drift
- [SharedPtr< Staircase >](#) s
- float ampl

#### 10.218.1 Detailed Description

A summary struct to handle task settings.

Definition at line 55 of file psycho-obfba.C.

The documentation for this struct was generated from the following files:

- [psycho-obfba.C](#)
- psycho-obfba2.C

## 10.219 PvmMessage Class Reference

definition and access functions for PVM messages

```
#include <pvmMessage.H>
```

### Public Member Functions

- [PvmMessage](#) ()  
*constructor for default PVM message encoding (PvmDataDefault):*
- [PvmMessage](#) (const int [encoding](#))  
*constructor with encoding type:*
- [~PvmMessage](#) ()  
*destructor:*
- void [initSend](#) ()  
*reinitialize a message for re-use:*
- void [send](#) (const int [tid](#), const int tag) const  
*send to host tid with given tag:*
- void [addImage](#) (const [Image](#)< [PixRGB](#)< [byte](#) > > &im)  
*add stuff to be sent out:*
- void [addFixation](#) (const int i, const int j, const int frame)  
*add stuff to be sent out:*
- void [addString](#) (const char \*s)  
*add stuff to be sent out:*
- int [receive](#) (const int [tid](#), const int tag, const int timeout=0)  
*receive a message (timeout in microsec)*
- int [getElement](#) ()  
*fetch next element type and return it:*
- void [getImage](#) ([Image](#)< [PixRGB](#)< [byte](#) > > &im)  
*unpack various elements which have been identified according to the value returned by [getElement\(\)](#):*
- void [getFixation](#) (int &i, int &j, int &frame)

*unpack various elements which have been identified according to the value returned by [getElement\(\)](#):*

- void [getString](#) (char \*s)  
*unpack various elements which have been identified according to the value returned by [getElement\(\)](#):*
- int [gettid](#) ()  
*return tid of sender from who we received a message:*
- int [getTag](#) ()  
*misc access functions:*

## Protected Member Functions

- void [getInfo](#) ()
- void [init](#) (const int [encoding](#))

## Protected Attributes

- int [bufid](#)  
*buffer ID (send & receive)*
- int [msgtag](#)  
*message identifier (send & receive)*
- int [tid](#)  
*tid message comes from (receive only)*
- int [info](#)  
*info about buffer (receive only)*
- int [siz](#)  
*buffer size (receive only)*
- int [encoding](#)  
*type of PVM low-level data encoding*

### 10.219.1 Detailed Description

definition and access functions for PVM messages

Definition at line 52 of file pvmMessage.H.

### 10.219.2 Constructor & Destructor Documentation

#### 10.219.2.1 PvmMessage::PvmMessage ()

constructor for default PVM message encoding (PvmDataDefault):

Definition at line 46 of file pvmMessage.C.

#### 10.219.2.2 PvmMessage::PvmMessage (const int *encoding*)

constructor with encoding type:

Definition at line 50 of file pvmMessage.C.

#### 10.219.2.3 PvmMessage::~PvmMessage ()

destructor:

Definition at line 54 of file pvmMessage.C.

### 10.219.3 Member Function Documentation

#### 10.219.3.1 void PvmMessage::addFixation (const int *i*, const int *j*, const int *frame*)

add stuff to be sent out:

Definition at line 89 of file pvmMessage.C.

References bufid, and initSend().

#### 10.219.3.2 void PvmMessage::addImage (const Image< PixRGB< byte > > & *im*)

add stuff to be sent out:

Definition at line 77 of file pvmMessage.C.

References bufid, and initSend().

**10.219.3.3 void PvmMessage::addString (const char \* s)**

add stuff to be sent out:

Definition at line 100 of file pvmMessage.C.

References bufid, and initSend().

**10.219.3.4 int PvmMessage::getElement ()**

fetch next element type and return it:

Definition at line 134 of file pvmMessage.C.

**10.219.3.5 void PvmMessage::getFixation (int & i, int & j, int & frame)**

unpack various elements which have been identified according to the value returned by [getElement\(\)](#):

Definition at line 148 of file pvmMessage.C.

**10.219.3.6 void PvmMessage::getImage (Image< PixRGB< byte > > & im)**

unpack various elements which have been identified according to the value returned by [getElement\(\)](#):

Definition at line 138 of file pvmMessage.C.

**10.219.3.7 void PvmMessage::getString (char \* s)**

unpack various elements which have been identified according to the value returned by [getElement\(\)](#):

Definition at line 152 of file pvmMessage.C.

**10.219.3.8 int PvmMessage::getTag ()**

misc access functions:

Definition at line 160 of file pvmMessage.C.

References msgtag.

**10.219.3.9 int PvmMessage::gettid ()**

return tid of sender from who we received a message:

Definition at line 156 of file pvmMessage.C.

References tid.

#### 10.219.3.10 void PvmMessage::initSend ()

reinitialize a message for re-use:

Definition at line 60 of file pvmMessage.C.

References bufid, and encoding.

Referenced by addFixation(), addImage(), and addString().

#### 10.219.3.11 int PvmMessage::receive (const int tid, const int tag, const int timeout = 0)

receive a message (timeout in microsec)

return value is the buffer ID or 0 if timed out:

Definition at line 109 of file pvmMessage.C.

References bufid.

#### 10.219.3.12 void PvmMessage::send (const int tid, const int tag) const

send to host tid with given tag:

Definition at line 66 of file pvmMessage.C.

References PVMSG\_END.

### 10.219.4 Member Data Documentation

#### 10.219.4.1 int PvmMessage::bufid [protected]

buffer ID (send & receive)

Definition at line 100 of file pvmMessage.H.

Referenced by addFixation(), addImage(), addString(), initSend(), and receive().

#### 10.219.4.2 int PvmMessage::encoding [protected]

type of PVM low-level data encoding

Definition at line 110 of file pvmMessage.H.



Referenced by `initSend()`.

#### 10.219.4.3 `int PvmMessage::info` [protected]

info about buffer (receive only)

Definition at line 106 of file `pvmMessage.H`.

#### 10.219.4.4 `int PvmMessage::msgtag` [protected]

message identifier (send & receive)

Definition at line 102 of file `pvmMessage.H`.

Referenced by `getTag()`.

#### 10.219.4.5 `int PvmMessage::siz` [protected]

buffer size (receive only)

Definition at line 108 of file `pvmMessage.H`.

#### 10.219.4.6 `int PvmMessage::tid` [protected]

tid message comes from (receive only)

Definition at line 104 of file `pvmMessage.H`.

Referenced by `gettid()`.

The documentation for this class was generated from the following files:

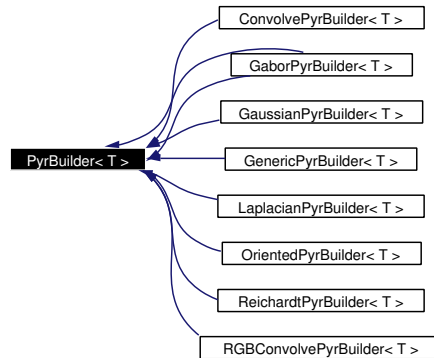
- [pvmMessage.H](#)
- [pvmMessage.C](#)

## 10.220 PyrBuilder< T > Class Template Reference

An interface class for creating dyadic pyramids from input images.

```
#include <PyrBuilder.H>
```

Inheritance diagram for PyrBuilder< T >:



### Public Member Functions

- [PyrBuilder](#) ()  
*Create an empty PyrBuilder.*
- virtual [~PyrBuilder](#) ()  
*Virtual destructor for safe inheritance.*
- virtual [PyrBuilder< T > \\* clone](#) () const=0  
*Cloning constructor useful to make clones from pointers:.*
- virtual [ImageSet< T > build](#) (const [Image< T > &image](#), const int depth)=0  
*Create a pyramid from an input image, with given depth (=nb levels).*
- virtual void [reset](#) ()  
*reset the pyramid - whatever that means for a specific pyramid*

### 10.220.1 Detailed Description

```
template<class T> class PyrBuilder< T >
```

An interface class for creating dyadic pyramids from input images.

A dyadic image pyramid is an array of images such that the next image is obtained from the previous one by applying some filter and decimating the image by a factor 2 horizontally and vertically.

The PyrBuilder classes essentially wrap one of the buildPyr\*() functions, binding pre-set values to all of the trailing arguments. This allows all of the buildPyr\*() functions to be presented behind a uniform interface.

PyrBuilder defers the actual handling of input images through the abstract function [build\(\)](#). Different subclasses may implement this to apply different filters (e.g., gaussian, laplacian, gabor) before decimation. [GenericPyrBuilder](#) offers a generic implementation, while [GaussianPyrBuilder](#), [LaplacianPyrBuilder](#), and [OrientedPyrBuilder](#) are specialized for single filter types.

Definition at line 65 of file PyrBuilder.H.

## 10.220.2 Constructor & Destructor Documentation

### 10.220.2.1 `template<class T> PyrBuilder< T >::PyrBuilder ()`

Create an empty PyrBuilder.

Definition at line 53 of file PyrBuilder.C.

### 10.220.2.2 `template<class T> PyrBuilder< T >::~~PyrBuilder () [virtual]`

Virtual destructor for safe inheritance.

Definition at line 58 of file PyrBuilder.C.

## 10.220.3 Member Function Documentation

### 10.220.3.1 `template<class T> virtual ImageSet<T> PyrBuilder< T >::build (const Image< T > & image, const int depth) [pure virtual]`

Create a pyramid from an input image, with given depth (=nb levels).

Implemented in [GaborPyrBuilder](#)< T >, [GaussianPyrBuilder](#)< T >, [ConvolvePyrBuilder](#)< T >, [RGBConvolvePyrBuilder](#)< T >, [LaplacianPyrBuilder](#)< T >, [OrientedPyrBuilder](#)< T >, [GenericPyrBuilder](#)< T >, [ReichardtPyrBuilder](#)< T >, [TemplateMatchPyrBuilder](#), [GaborPyrBuilder](#)< T >, and [GaussianPyrBuilder](#)< PixRGB< byte > >.

**10.220.3.2** `template<class T> virtual PyrBuilder<T>* PyrBuilder< T>::clone () const [pure virtual]`

Cloning constructor useful to make clones from pointers:.

Implemented in [GaborPyrBuilder< T >](#), [GaussianPyrBuilder< T >](#), [ConvolvePyrBuilder< T >](#), [RGBConvolvePyrBuilder< T >](#), [LaplacianPyrBuilder< T >](#), [OrientedPyrBuilder< T >](#), [GenericPyrBuilder< T >](#), [ReichardtPyrBuilder< T >](#), [TemplateMatchPyrBuilder](#), [GaborPyrBuilder< T >](#), and [GaussianPyrBuilder< PixRGB< byte > >](#).

**10.220.3.3** `template<class T> void PyrBuilder< T >::reset () [virtual]`

reset the pyramid - whatever that means for a specific pyramid

a no op implementation is given in this base class - so if pyramids don't need this, they don't have to worry about it

Reimplemented in [ReichardtPyrBuilder< T >](#).

Definition at line 63 of file [PyrBuilder.C](#).

The documentation for this class was generated from the following files:

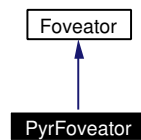
- [PyrBuilder.H](#)
- [PyrBuilder.C](#)

## 10.221 PyrFoveator Class Reference

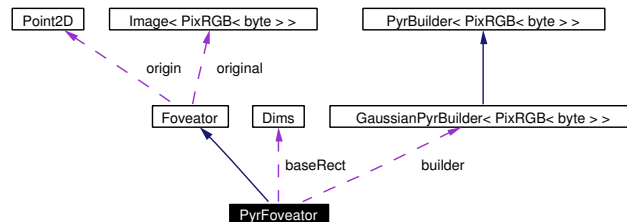
A class that does space-variant processing with dyadic pyramid.

```
#include <PyrFoveator.H>
```

Inheritance diagram for PyrFoveator:



Collaboration diagram for PyrFoveator:



### Public Member Functions

- **PyrFoveator** (const **Image**< **PixRGB**< byte > > &img, int filterSize)  
*Construct a PyrFoveator for img.*
- **PyrFoveator** (const **PyrFoveator** &pf)  
*Construct a PyrFoveator from another.*
- **PyrFoveator** & **operator=** (const **PyrFoveator** &pf)  
*Construct a PyrFoveator equal to another.*
- **~PyrFoveator** ()  
*Destructor.*
- void **setBaseRect** (**Dims** &d)  
*Set dimensions of smallest rectangle with highest resolution.*

- void [setBaseRect](#) (int x, int y)  
*Set dimensions of smallest rectangle with highest resolution.*
- [Image](#)< [PixRGB](#)< [byte](#) > > [foveate](#) (void)  
*Perform dyadic pyramid foveation and return resulting image.*

## Static Public Member Functions

- [Image](#)< [PixRGB](#)< [byte](#) > > [foveate](#) (const [Image](#)< [PixRGB](#)< [byte](#) > > &img, int filterSize, int baseRectWidth, int baseRectHeight, int x, int y)  
*Perform one dyadic pyramid foveation without a [PyrFoveator](#).*

### 10.221.1 Detailed Description

A class that does space-variant processing with dyadic pyramid.

A Gaussian pyramid is constructed from the original image, each level with resolution halved, image size 1/4 of previous level. After building the pyramid, a foveated image is reassembled by starting with low-resolution image expanded to full-size, then superimposing smaller rectangles of higher resolution.

Testing has shown that [PyrFoveator](#) is somewhat faster than [BlurFoveator](#) and can perform similar blurring, but [PyrFoveator](#) creates noticeable straight-line boundaries between rectangles of differing resolution.

Definition at line 21 of file [PyrFoveator.H](#).

### 10.221.2 Constructor & Destructor Documentation

#### 10.221.2.1 [PyrFoveator::PyrFoveator](#) (const [Image](#)< [PixRGB](#)< [byte](#) > > & *img*, int *filterSize*)

Construct a [PyrFoveator](#) for *img*.

This constructor will create a [PyrFoveator](#) that handles *img* and all other images of the same dimensions.

#### Parameters:

- img* image to be foveated
- filterSize* size of blur filter to be applied

Definition at line 10 of file [PyrFoveator.C](#).

Referenced by [operator=\(\)](#).

**10.221.2.2 PyrFoveator::PyrFoveator (const [PyrFoveator](#) & pf)**

Construct a PyrFoveator from another.

Definition at line 18 of file PyrFoveator.C.

References [baseRect](#).

**10.221.2.3 PyrFoveator::~~[PyrFoveator](#) ()**

Destructor.

Definition at line 32 of file PyrFoveator.C.

**10.221.3 Member Function Documentation****10.221.3.1 [Image](#)< [PixRGB](#)< [byte](#) > > PyrFoveator::foveate (const [Image](#)< [PixRGB](#)< [byte](#) > > & img, int filterSize, int baseRectWidth, int baseRectHeight, int x, int y) [static]**

Perform one dyadic pyramid foveation without a PyrFoveator.

This function is for foveating ONE image with a call such as [PyrFoveator::foveate\(\)](#). It is not optimized for foveating multiple images.

**Parameters:**

- img* image to be foveated
- filterSize* size of blur filter to be applied
- baseRectWidth* width of base rectangle
- baseRectHeight* height of base rectangle
- x* horizontal coordinate of origin
- y* vertical coordinate of origin

Definition at line 102 of file PyrFoveator.C.

References [GaussianPyrBuilder](#)< T >::build(), [Image](#)< T >::coordsOk(), [rescale\(\)](#), and [Image](#)< T >::setVal().

**10.221.3.2 [Image](#)< [PixRGB](#)< [byte](#) > > PyrFoveator::foveate (void) [virtual]**

Perform dyadic pyramid foveation and return resulting image.

Implements [Foveator](#).

Definition at line 53 of file PyrFoveator.C.

References `GaussianPyrBuilder< PixRGB< byte > >::build()`, `Image< T >::coordsOk()`, `Dims::h()`, `Point2D::i`, `Point2D::j`, `rescale()`, `Image< T >::setVal()`, and `Dims::w()`.

Referenced by `main()`.

#### 10.221.3.3 [PyrFoveator](#) & `PyrFoveator::operator= (const PyrFoveator & pf)`

Construct a `PyrFoveator` equal to another.

Definition at line 25 of file `PyrFoveator.C`.

References `PyrFoveator()`.

#### 10.221.3.4 `void PyrFoveator::setBaseRect (int x, int y)`

Set dimensions of smallest rectangle with highest resolution.

Definition at line 45 of file `PyrFoveator.C`.

#### 10.221.3.5 `void PyrFoveator::setBaseRect (Dims & d)`

Set dimensions of smallest rectangle with highest resolution.

Definition at line 40 of file `PyrFoveator.C`.

Referenced by `main()`.

The documentation for this class was generated from the following files:

- [PyrFoveator.H](#)
- [PyrFoveator.C](#)

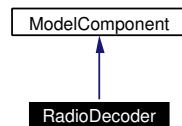


## 10.222 RadioDecoder Class Reference

This class runs a thread which records servo position from sound card.

```
#include <RadioDecoder.H>
```

Inheritance diagram for RadioDecoder:



Collaboration diagram for RadioDecoder:

### Public Member Functions

- `RadioDecoder (ModelManager &mgr, const std::string &descrName="R/C Radio Decoder", const std::string &tagName="RadioDecoder")`

*Constructor.*

- `~RadioDecoder ()`

*Destructor.*

- virtual void `exportOptions` (const int whatToExport, const bool recurse=true)

*Export some of our `ModelParam` parameters as command-line-options.*

- void `zeroCalibrate` (const int nbiter=20)

*Calibrate the zero (will block until done).*

- void `rangeCalibrate` (const int nbiter=200)

*Calibrate the positive and negative ranges to +/-1.0.*

- float `getVal` (const int channel) const

*Access function; values guaranteed within ]-1.0 .. 1.0[.*

- void `run` ()

*This method should not be called (implements thread code).*

## Protected Member Functions

- void [start1](#) ()  
*get started (see [ModelComponent.H](#))*
- void [start2](#) ()  
*get started (see [ModelComponent.H](#))*
- void [stop1](#) ()  
*get stopped (see [ModelComponent.H](#))*

## Protected Attributes

- [ModelParam](#)< float > \* [zero](#)  
*zero calibration values*
- [ModelParam](#)< float > \* [posmult](#)  
*positive multiplier calibration values*
- [ModelParam](#)< float > \* [negmult](#)  
*negative multiplier calibration values*

### 10.222.1 Detailed Description

This class runs a thread which records servo position from sound card.

Radio-control servomechanisms are controlled by a pulse-width-modulated protocol in which a 1-2ms pulse (from 0V to 4.8V) is generated every 18-25ms. The width of the pulse (between 1 and 2ms) determines the position of the servo. This class uses an [AudioGrabber](#) to record pulses, for example coming from an R/C receiver. It then computes the pulse width and converts it to a number between -1.0 and 1.0. The current pulse values are available via [getVal\(\)](#), which is a non-blocking function call.

Definition at line 56 of file RadioDecoder.H.

### 10.222.2 Constructor & Destructor Documentation

**10.222.2.1** [RadioDecoder::RadioDecoder](#) ([ModelManager](#) & *mgr*, const [std::string](#) & *descrName* = "R/C Radio Decoder", const [std::string](#) & *tagName* = "RadioDecoder")

Constructor.

Definition at line 67 of file RadioDecoder.C.

References `ModelComponent::addSubComponent()`, and `uint`.

#### 10.222.2.2 RadioDecoder::~RadioDecoder ()

Destructor.

Definition at line 156 of file RadioDecoder.C.

### 10.222.3 Member Function Documentation

#### 10.222.3.1 void RadioDecoder::exportOptions (const int *whatToExport*, const bool *recurse* = true) [virtual]

Export some of our [ModelParam](#) parameters as command-line-options.

See the base function in [ModelComponent.H](#) for info.

Reimplemented from [ModelComponent](#).

Definition at line 87 of file RadioDecoder.C.

References `ModelComponent::doRequestOption()`, and `OPTEXP_CORE`.

#### 10.222.3.2 float RadioDecoder::getVal (const int *channel*) const

Access function; values guaranteed within  $]-1.0 .. 1.0[$ .

Definition at line 160 of file RadioDecoder.C.

References `ASSERT`, `ModelParam< float >::getVal()`, `negmult`, and `posmult`.

Referenced by `rangeCalibrate()`.

#### 10.222.3.3 void RadioDecoder::rangeCalibrate (const int *nbiter* = 200)

Calibrate the positive and negative ranges to  $\pm 1.0$ .

Definition at line 199 of file RadioDecoder.C.

References `getVal()`, `negmult`, `posmult`, and `ModelParam< float >::setVal()`.

#### 10.222.3.4 void RadioDecoder::run ()

This method should not be called (implements thread code).

Definition at line 237 of file RadioDecoder.C.

**10.222.3.5 void RadioDecoder::start1 () [protected, virtual]**

get started (see [ModelComponent.H](#))

Reimplemented from [ModelComponent](#).

Definition at line 116 of file RadioDecoder.C.

References [ModelParam< float >::init\(\)](#), [negmult](#), and [posmult](#).

**10.222.3.6 void RadioDecoder::start2 () [protected, virtual]**

get started (see [ModelComponent.H](#))

Reimplemented from [ModelComponent](#).

Definition at line 136 of file RadioDecoder.C.

**10.222.3.7 void RadioDecoder::stop1 () [protected, virtual]**

get stopped (see [ModelComponent.H](#))

Reimplemented from [ModelComponent](#).

Definition at line 143 of file RadioDecoder.C.

References [negmult](#), and [posmult](#).

**10.222.3.8 void RadioDecoder::zeroCalibrate (const int *nbiter* = 20)**

Calibrate the zero (will block until done).

Definition at line 180 of file RadioDecoder.C.

References [ModelParam< float >::setVal\(\)](#).

**10.222.4 Member Data Documentation****10.222.4.1 [ModelParam<float>\\*](#) [RadioDecoder::negmult](#) [protected]**

negative multiplier calibration values

Definition at line 86 of file RadioDecoder.H.

Referenced by [getVal\(\)](#), [rangeCalibrate\(\)](#), [start1\(\)](#), and [stop1\(\)](#).

**10.222.4.2** [ModelParam<float>\\*](#) [RadioDecoder::posmult](#) [protected]

positive multiplier calibration values

Definition at line 85 of file RadioDecoder.H.

Referenced by [getVal\(\)](#), [rangeCalibrate\(\)](#), [start1\(\)](#), and [stop1\(\)](#).

**10.222.4.3** [ModelParam<float>\\*](#) [RadioDecoder::zero](#) [protected]

zero calibration values

Definition at line 84 of file RadioDecoder.H.

The documentation for this class was generated from the following files:

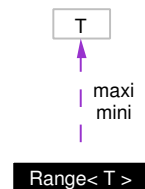
- [RadioDecoder.H](#)
- [RadioDecoder.C](#)

## 10.223 Range< T > Class Template Reference

A class for managing min/max range information.

```
#include <Range.H>
```

Collaboration diagram for Range< T >:



### Public Member Functions

- [Range](#) ()  
*Default constructor.*
- [Range](#) (const T &mi, const T &ma)  
*Construct with given min and max values.*
- const T & [min](#) () const  
*Get the current min value.*
- const T & [max](#) () const  
*Get the current max value.*
- T [range](#) () const  
*Get the range (i.e. the difference of max minus min).*
- void [setMin](#) (const T &mi)  
*Set the range's min value.*
- void [setMax](#) (const T &ma)  
*Set the range's max value.*
- void [mergeMin](#) (const T &v)  
*Update our min value if val is less than the current max.*
- void [mergeMax](#) (const T &val)

*Update our max value if val is greater than the current max.*

- void [merge](#) (const T &val)

*Update our min or max value if val falls outside of the current range.*

- void [operator\(\)](#) (const T &v)

*This operator() allows straightforward use in generic algorithms.*

- void [merge](#) (const [Range](#)< T > &other)

*Merge other's min and max values into ours.*

### 10.223.1 Detailed Description

**template<class T> class Range< T >**

A class for managing min/max range information.

This class is primarily intended to work with algorithms that manipulate the dynamic range of Image<T>'s. For example, it can help to specify how an image should have its current dynamic range remapped to a new range (see [remapRange\(\)](#) in [Image\\_MathOps.H](#)). In addition, the Range<T> class provides an operator() that allows it to be used as a functor in generic algorithms for accumulating the min/max over a sequence of elements (as in [rangeOf\(\)](#) in [Image\\_MathOps.H](#)).

Definition at line 49 of file Range.H.

### 10.223.2 Constructor & Destructor Documentation

**10.223.2.1 template<class T> [Range](#)< T >::Range () [inline]**

Default constructor.

We initialize the object's min element with type T's **max** value, and we initialize the object's max element with type T's **min** value. This seeming paradoxical situation is in fact "The Right Thing" if we want to accumulate the min/max over a range of elements. The starting condition of the Range object should make the weakest possible assumption about the values that will be observed; and this weakest assumption is precisely to start with the min being as large as possible, and the max being as small as possible.

Definition at line 116 of file Range.H.

**10.223.2.2** `template<class T> Range< T >::Range (const T & mi, const T & ma) [inline]`

Construct with given min and max values.

Definition at line 123 of file Range.H.

### 10.223.3 Member Function Documentation

**10.223.3.1** `template<class T> const T& Range< T >::max (void) const [inline]`

Get the current max value.

Definition at line 73 of file Range.H.

Referenced by `doOneNormalize()`, `Range< T >::merge()`, and `normalizeWithScale()`.

**10.223.3.2** `template<class T> void Range< T >::merge (const Range< T > & other) [inline]`

Merge other's min and max values into ours.

Definition at line 152 of file Range.H.

References `Range< T >::max()`, `Range< T >::mergeMax()`, `Range< T >::mergeMin()`, and `Range< T >::min()`.

**10.223.3.3** `template<class T> void Range< T >::merge (const T & val) [inline]`

Update our min or max value if val falls outside of the current range.

Definition at line 137 of file Range.H.

References `Range< T >::mergeMax()`, and `Range< T >::mergeMin()`.

Referenced by `Range< T >::operator()()`, and `rangeOf()`.

**10.223.3.4** `template<class T> void Range< T >::mergeMax (const T & val) [inline]`

Update our max value if val is greater than the current max.

Definition at line 133 of file Range.H.

Referenced by `Range< T >::merge()`.



**10.223.3.5** `template<class T> void Range< T >::mergeMin (const T & v)`  
`[inline]`

Update our min value if val is less than the current max.

Definition at line 129 of file Range.H.

Referenced by `Range< T >::merge()`.

**10.223.3.6** `template<class T> const T& Range< T >::min () const`  
`[inline]`

Get the current min value.

Definition at line 70 of file Range.H.

Referenced by `Range< T >::merge()`, `normalizeWithScale()`, and `remapRange()`.

**10.223.3.7** `template<class T> void Range< T >::operator() (const T & v)`  
`[inline]`

This operator() allows straightforward use in generic algorithms.

In particular, this is useful for accumulating the min/max over a range of elements. For example, `rangeOf()` could be implemented like this:

```
Image<T> img;  
Range<T> rng = std::for_each(img.begin(), img.end(), Range<T>());
```

We construct a default `Range<T>` object, which is copied and passed to `for_each()`, which in turn calls `operator()` on that range object for each element in the iterator sequence that is visited.

Definition at line 145 of file Range.H.

References `Range< T >::merge()`.

**10.223.3.8** `template<class T> T Range< T >::range () const` `[inline]`

Get the range (i.e. the difference of max minus min).

Definition at line 76 of file Range.H.

Referenced by `remapRange()`.

**10.223.3.9** `template<class T> void Range< T >::setMax (const T & ma)`  
[inline]

Set the range's max value.

Definition at line 82 of file Range.H.

**10.223.3.10** `template<class T> void Range< T >::setMin (const T & mi)`  
[inline]

Set the range's min value.

Definition at line 79 of file Range.H.

The documentation for this class was generated from the following file:

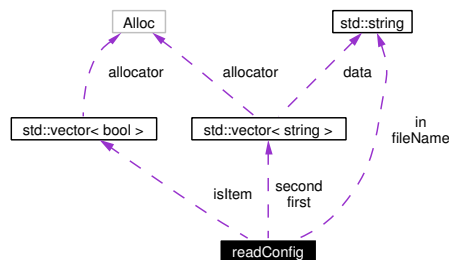
- [Range.H](#)

## 10.224 readConfig Class Reference

open a config file and parse it for data retrieval

```
#include <readConfig.H>
```

Collaboration diagram for readConfig:



### Public Member Functions

- `readConfig ()`  
*constructor with initial size = 25*
- `readConfig (int size)`  
*constructor with initial size*
- `~readConfig ()`  
*destructor for readConfig*
- `void openFile (char *filename, bool echo=false)`  
*open a config file with filename*
- `void writeFile ()`  
*write the file back to disk as filename*
- `void writeFile (const char *filename)`  
*write the file back to disk with a new file name*
- `bool readFileTrue (int itemNumber)`  
*returns true if item number itemNumber exists*
- `int readFileValueI (int itemNumber)`  
*return value of item as an integer*

- float `readFileValueF` (int `itemNumber`)  
*return value of item as a float*
- string `readFileValueS` (int `itemNumber`)  
*return value of item as a string*
- const char \* `readFileValueC` (int `itemNumber`)  
*return value of item as a C style char\**
- string `readFileValueName` (int `itemNumber`)  
*return the item name*
- const char \* `readFileValueNameC` (int `itemNumber`)  
*return item name as a C style char\**
- float `readFileValueNameF` (int `itemNumber`)  
*return item name as a float*
- float `getItemValueF` (string `itemName`)  
*return the float value of a number with name itemName*
- string `getItemValueS` (string `itemName`)  
*return the string value of a number with name itemName*
- bool `getItemValueB` (string `itemName`)  
*return a bool representation of the value 1 = true else false*
- const char \* `getItemValueC` (string `itemName`)  
*return the char\* value of a number with name itemName*
- void `setItemValue` (string `itemName`, float `_set`)  
*Set the item with string name itemName to a float \_set.*
- void `setItemValue` (string `itemName`, string `_set`)  
*Set the item with string name itemName to a string \_set.*
- void `setItemValue` (string `itemName`, const char \* `_set`)  
*Set the item with string name itemName to a const char\* \_set.*
- void `setFileValue` (int `itemNumber`, float `_set`)  
*Set the item signified by its number to a float.*

- void `setFileValue` (int `itemNumber`, string `_set`)  
*Set the item signified by its number to a string.*
- void `setFileValue` (int `itemNumber`, const char \*`_set`)  
*Set the item signified by its number to a const char\*.*
- int `addItemValue` (string `itemName`, float `_set`)  
*Set the item with string name `itemName` to a float `_set`.*
- int `addItemValue` (string `itemName`, string `_set`)  
*Set the item with string name `itemName` to a string `_set`.*
- int `addItemValue` (string `itemName`, const char \*`_set`)  
*Set the item with string name `itemName` to a string `_set`.*
- int `itemCount` ()  
*returns how many items exist*

### 10.224.1 Detailed Description

open a config file and parse it for data retrieval

This class works to open a file for parsing which has name value key pairs. Each pair is separated and loaded into a vector. Values can then be retrieved by either calling a values item number or name. This will then return the value. Comments can be handled in the config file so long as they are in the form '# words #'. That is place a # with a space before and after each comment.

Definition at line 68 of file `readConfig.H`.

### 10.224.2 Constructor & Destructor Documentation

#### 10.224.2.1 `readConfig::readConfig ()`

constructor with initial size = 25

Definition at line 43 of file `readConfig.C`.

#### 10.224.2.2 `readConfig::readConfig (int size)`

constructor with initial size

Definition at line 52 of file `readConfig.C`.

**10.224.2.3 readConfig::~~readConfig ()**

destructor for readConfig

Definition at line 60 of file readConfig.C.

**10.224.3 Member Function Documentation****10.224.3.1 int readConfig::addItemValue (string itemName, const char \* \_set)**

Set the item with string name itemName to a string \_set.

Definition at line 349 of file readConfig.C.

References addItemValue().

**10.224.3.2 int readConfig::addItemValue (string itemName, string \_set)**

Set the item with string name itemName to a string \_set.

Definition at line 332 of file readConfig.C.

**10.224.3.3 int readConfig::addItemValue (string itemName, float \_set)**

Set the item with string name itemName to a float \_set.

Definition at line 324 of file readConfig.C.

Referenced by addItemValue().

**10.224.3.4 bool readConfig::getItemValueB (string itemName)**

return a bool representation of the value 1 = true else false

Definition at line 227 of file readConfig.C.

References ASSERT.

**10.224.3.5 const char \* readConfig::getItemValueC (string itemName)**

return the char\* value of a number with name itemName

Definition at line 245 of file readConfig.C.

References ASSERT.

**10.224.3.6 float readConfig::getItemValueF ([string](#) itemName)**

return the float value of a number with name itemName

Definition at line 197 of file readConfig.C.

References ASSERT.

Referenced by ContourNeuronCreate< TC2 >::CreateNeuron(), main(), and Omni< T >::run().

**10.224.3.7 [string](#) readConfig::getItemValueS ([string](#) itemName)**

return the string value of a number with name itemName

Definition at line 212 of file readConfig.C.

References ASSERT.

**10.224.3.8 int readConfig::itemCount ()**

returns how many items exist

Definition at line 355 of file readConfig.C.

**10.224.3.9 void readConfig::openFile (char \* *filename*, bool *echo* = false)**

open a config file with filename

Definition at line 64 of file readConfig.C.

Referenced by cameraConf::cameraConf(), carConf::carConf(), and main().

**10.224.3.10 bool readConfig::readFileTrue (int *itemNumber*)**

returns true if item number itemNumber exists

Definition at line 143 of file readConfig.C.

References itemNumber.

Referenced by main().

**10.224.3.11 const char \* readConfig::readFileValueC (int *itemNumber*)**

return value of item as a C style char\*

Definition at line 173 of file readConfig.C.

References ASSERT, and itemNumber.

#### **10.224.3.12 float readConfig::readFileValueF (int *itemNumber*)**

return value of item as a float

Definition at line 161 of file readConfig.C.

References ASSERT, and itemNumber.

Referenced by main().

#### **10.224.3.13 int readConfig::readFileValueI (int *itemNumber*)**

return value of item as an integer

Definition at line 155 of file readConfig.C.

References ASSERT, and itemNumber.

#### **10.224.3.14 string readConfig::readFileValueName (int *itemNumber*)**

return the item name

Definition at line 179 of file readConfig.C.

References ASSERT, and itemNumber.

Referenced by main().

#### **10.224.3.15 const char \* readConfig::readFileValueNameC (int *itemNumber*)**

return item name as a C style char\*

Definition at line 185 of file readConfig.C.

References ASSERT, and itemNumber.

#### **10.224.3.16 float readConfig::readFileValueNameF (int *itemNumber*)**

return item name as a float

Definition at line 191 of file readConfig.C.

References ASSERT, and itemNumber.



**10.224.3.17** [string](#) readConfig::readFileValuesS (int *itemNumber*)

return value of item as a string

Definition at line 167 of file readConfig.C.

References ASSERT, and itemNumber.

**10.224.3.18** void readConfig::setFileValue (int *itemNumber*, const char \* *\_set*)

Set the item signified by its number to a const char\*.

Definition at line 318 of file readConfig.C.

References ASSERT, and itemNumber.

**10.224.3.19** void readConfig::setFileValue (int *itemNumber*, [string](#) *\_set*)

Set the item signified by its number to a string.

Definition at line 312 of file readConfig.C.

References ASSERT, and itemNumber.

**10.224.3.20** void readConfig::setFileValue (int *itemNumber*, float *\_set*)

Set the item signified by its number to a float.

Definition at line 304 of file readConfig.C.

References ASSERT, and itemNumber.

**10.224.3.21** void readConfig::setItemValue ([string](#) *itemName*, const char \* *\_set*)

Set the item with string name itemName to a const char\* \_set.

Definition at line 290 of file readConfig.C.

References ASSERT.

**10.224.3.22** void readConfig::setItemValue ([string](#) *itemName*, [string](#) *\_set*)

Set the item with string name itemName to a string \_set.

Definition at line 276 of file readConfig.C.

References ASSERT.

**10.224.3.23 void readConfig::setItemValue ([string](#) *itemName*, float *\_set*)**

Set the item with string name *itemName* to a float *\_set*.

Definition at line 260 of file readConfig.C.

References ASSERT.

**10.224.3.24 void readConfig::writeFile (const char \* *filename*)**

write the file back to disk with a new file name

Definition at line 130 of file readConfig.C.

**10.224.3.25 void readConfig::writeFile ()**

write the file back to disk as filename

Definition at line 117 of file readConfig.C.

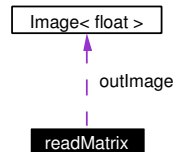
The documentation for this class was generated from the following files:

- [readConfig.H](#)
- [readConfig.C](#)

## 10.225 readMatrix Class Reference

```
#include <readMatrix.H>
```

Collaboration diagram for readMatrix:



### Public Member Functions

- **readMatrix** (const char \*fileName)
- **std::vector< std::vector< double > > returnMatrix** ()
- **Image< float > returnMatrixAsImage** ()
- void **echoMatrix** ()

### 10.225.1 Detailed Description

this class will read in the file fileName as a standard whitespace delimited file and put it into a matrix. The matrix can be retrieved in either a double nested vector<float> or an **Image<float>**. The input file must state the dimensions on the first line. Comments may be placed using # characters

Definition at line 23 of file readMatrix.H.

The documentation for this class was generated from the following files:

- readMatrix.H
- readMatrix.C

## 10.226 Rectangle Class Reference

A basic rectangle class.

```
#include <Rectangle.H>
```

### Public Member Functions

- [Rectangle](#) (int tt, int ll, int bb, int rr)  
*Build from 4 coordinates (top, left, bottom, right).*
- [Rectangle](#) (int tt, int ll, int bb, int rr, [Dims](#) dims)  
*Build from 4 coordinates and adapt to fit into dims.*
- [Rectangle](#) (const [Point2D](#) &topleft, const [Dims](#) &dims)  
*Build from a top-left corner and some dims.*
- [Rectangle](#) (const [Rectangle](#) &rect)  
*Copy constructor.*
- [Rectangle](#) ()  
*Uninitialized constructor (useful for arrays).*
- void [fitToDims](#) (const [Dims](#) dims)  
*modify the rectangle so that it fits within an image of dimensions dims*
- void [fitToRect](#) (const [Rectangle](#) &rect)  
*modify the rectangle so that it fits within the rectangle rect*
- void [modify](#) (int tt, int ll, int bb, int rr, [Dims](#) dims=[Dims](#)(0, 0))  
*Change coordinates and rebuild and fit into dims.*
- int [top](#) () const  
*Access internal components.*
- int [bottom](#) () const  
*Access internal components.*
- int [left](#) () const  
*Access internal components.*
- int [right](#) () const

*Access internal components.*

- int [width](#) () const

*Access some metric info.*

- int [height](#) () const

*Access some metric info.*

- bool [isValid](#) () const

*whether the rectangle is valid*

### 10.226.1 Detailed Description

A basic rectangle class.

This is a shorthand class to represent a 2D rectangle, as defined by integer top, left, bottom and right coordinates. The goal of this object is not to have to check if left < right, etc... for each function using a Rectangle. The object checks for that at its creation, and is always internally coherent. [Image](#) convention: top < bottom, left < right. This class is 100% inlined, so there is no .C file.

Definition at line 53 of file Rectangle.H.

### 10.226.2 Constructor & Destructor Documentation

#### 10.226.2.1 [Rectangle::Rectangle \(int tt, int ll, int bb, int rr\)](#) [inline]

Build from 4 coordinates (top, left, bottom, right).

Definition at line 118 of file Rectangle.H.

References [modify\(\)](#).

#### 10.226.2.2 [Rectangle::Rectangle \(int tt, int ll, int bb, int rr, \[Dims\]\(#\) dims\)](#) [inline]

Build from 4 coordinates and adapt to fit into dims.

Definition at line 122 of file Rectangle.H.

References [modify\(\)](#).

### 10.226.2.3 **Rectangle::Rectangle** (const [Point2D](#) & *topleft*, const [Dims](#) & *dims*) [inline]

Build from a top-left corner and some dims.

Definition at line 126 of file Rectangle.H.

References [Dims::h\(\)](#), [Point2D::i](#), [Point2D::j](#), [modify\(\)](#), and [Dims::w\(\)](#).

### 10.226.2.4 **Rectangle::Rectangle** (const [Rectangle](#) & *rect*) [inline]

Copy constructor.

Definition at line 131 of file Rectangle.H.

### 10.226.2.5 **Rectangle::Rectangle** () [inline]

Uninitialized constructor (useful for arrays).

Use [modify\(\)](#) to change Rectangle contents later

Definition at line 136 of file Rectangle.H.

Referenced by [fitToDims\(\)](#).

## 10.226.3 Member Function Documentation

### 10.226.3.1 **int Rectangle::bottom** () const [inline]

Access internal components.

Definition at line 174 of file Rectangle.H.

References [ASSERT](#).

Referenced by [convertToString\(\)](#), [copy\(\)](#), [BitObject::doesIntersect\(\)](#), [BitObject::drawBoundingBox\(\)](#), [BitObject::drawOutline\(\)](#), [Image< T >::drawRect\(\)](#), [BitObject::drawShape\(\)](#), [extractBitObjects\(\)](#), [fitToRect\(\)](#), [SimulationViewerCompress::getTraj\(\)](#), [Image< T >::rectangleOk\(\)](#), [BitObject::setMaxMinAvgIntensity\(\)](#), and [BitObject::writeToStream\(\)](#).

### 10.226.3.2 **void Rectangle::fitToDims** (const [Dims](#) *dims*) [inline]

modify the rectangle so that it fits within an image of dimensions dims

Definition at line 141 of file Rectangle.H.

References ASSERT, fitToRect(), Dims::h(), Dims::isNonEmpty(), Rectangle(), and Dims::w().

Referenced by extractBitObjects(), RetinaStd::getRawInputRectangle(), Retina::getRawInputRectangle(), SimulationViewerCompress::getTraj(), and modify().

### 10.226.3.3 void Rectangle::fitToRect (const Rectangle & rect) [inline]

modify the rectangle so that it fits within the rectangle rect

Definition at line 148 of file Rectangle.H.

References bottom(), left(), right(), and top().

Referenced by fitToDims().

### 10.226.3.4 int Rectangle::height () const [inline]

Access some metric info.

Definition at line 186 of file Rectangle.H.

References ASSERT.

Referenced by Image< T >::embed(), BitObject::getBoundingBox(), and Mbari-ResultViewer::saveSingleEventFrame().

### 10.226.3.5 bool Rectangle::isValid () const [inline]

whether the rectangle is valid

Definition at line 189 of file Rectangle.H.

Referenced by BitObject::isValid(), and BitObject::writeToStream().

### 10.226.3.6 int Rectangle::left () const [inline]

Access internal components.

Definition at line 177 of file Rectangle.H.

References ASSERT.

Referenced by convertToString(), copy(), BitObject::doesIntersect(), BitObject::drawBoundingBox(), BitObject::drawOutline(), Image< T >::drawRect(), BitObject::drawShape(), Image< T >::embed(), extractBitObjects(), TemplateMatchChannel::findBestMatch(), fitToRect(), BitObject::getObjectOrigin(), SimulationViewerCompress::getTraj(), Image< T >::rectangleOk(), BitObject::reset(), Mbari-

ResultViewer::saveSingleEventFrame(), BitObject::setMaxMinAvgIntensity(), FOEstimator::updateFOE(), and BitObject::writeToStream().

#### 10.226.3.7 void Rectangle::modify (int tt, int ll, int bb, int rr, Dims dims = Dims(0, 0)) [inline]

Change coordinates and rebuild and fit into dims.

Definition at line 161 of file Rectangle.H.

References fitToDims(), and Dims::isNonEmpty().

Referenced by centroid(), convertFromString(), SimulationViewerStd::drawMega-Combo(), VisualEventSet::drawTokens(), floodCleanBB(), main(), Rectangle(), and BitObject::reset().

#### 10.226.3.8 int Rectangle::right () const [inline]

Access internal components.

Definition at line 180 of file Rectangle.H.

References ASSERT.

Referenced by convertToString(), copy(), BitObject::doesIntersect(), BitObject::drawBoundingBox(), BitObject::drawOutline(), Image< T >::drawRect(), BitObject::drawShape(), extractBitObjects(), fitToRect(), SimulationViewerCompress::getTraj(), Image< T >::rectangleOk(), BitObject::setMaxMinAvgIntensity(), and BitObject::writeToStream().

#### 10.226.3.9 int Rectangle::top () const [inline]

Access internal components.

Definition at line 171 of file Rectangle.H.

References ASSERT.

Referenced by convertToString(), copy(), BitObject::doesIntersect(), BitObject::drawBoundingBox(), BitObject::drawOutline(), Image< T >::drawRect(), BitObject::drawShape(), Image< T >::embed(), extractBitObjects(), TemplateMatchChannel::findBestMatch(), fitToRect(), BitObject::getObjectOrigin(), SimulationViewerCompress::getTraj(), Image< T >::rectangleOk(), BitObject::reset(), Mbari-ResultViewer::saveSingleEventFrame(), BitObject::setMaxMinAvgIntensity(), FOEstimator::updateFOE(), and BitObject::writeToStream().



**10.226.3.10 int Rectangle::width () const [inline]**

Access some metric info.

Definition at line 183 of file Rectangle.H.

References ASSERT.

Referenced by Image< T >::embed(), BitObject::getBoundingBox(), and Mbari-ResultViewer::saveSingleEventFrame().

The documentation for this class was generated from the following file:

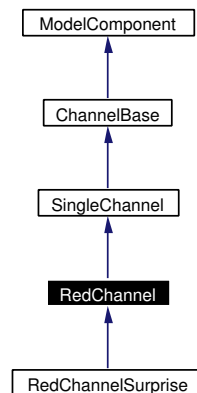
- [Rectangle.H](#)

## 10.227 RedChannel Class Reference

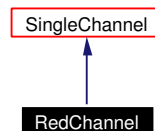
A red color channel.

```
#include <Channels.H>
```

Inheritance diagram for RedChannel:



Collaboration diagram for RedChannel:



### Public Member Functions

- [RedChannel](#) ([ModelManager](#) &mgr)  
*Construct with standard params.*
- virtual [~RedChannel](#) ()  
*Destructor.*

### 10.227.1 Detailed Description

A red color channel.

Definition at line 184 of file Channels.H.

## 10.227.2 Constructor & Destructor Documentation

### 10.227.2.1 RedChannel::RedChannel ([ModelManager](#) & mgr)

Construct with standard params.

Definition at line 229 of file Channels.C.

References [makeShared\(\)](#), [R](#), and [ModelParam< bool >::setVal\(\)](#).

### 10.227.2.2 RedChannel::~~RedChannel () [virtual]

Destructor.

Definition at line 236 of file Channels.C.

The documentation for this class was generated from the following files:

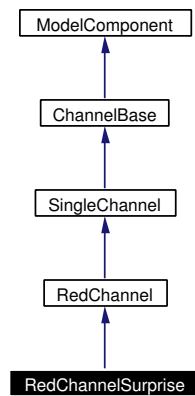
- [Channels.H](#)
- [Channels.C](#)

## 10.228 RedChannelSurprise Class Reference

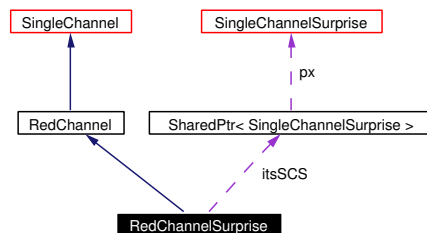
A red color channel.

```
#include <ChannelsSurprise.H>
```

Inheritance diagram for RedChannelSurprise:



Collaboration diagram for RedChannelSurprise:



### Public Member Functions

- [RedChannelSurprise](#) ([ModelManager](#) &mgr)  
*Construct with standard params.*
- virtual [~RedChannelSurprise](#) ()  
*Destructor.*
- virtual [Image](#)< float > [computeSubmap](#) (const [uint](#) index) const

*computeSubmap()* is special and will return surprise

## Protected Attributes

- [SharedPtr< SingleChannelSurprise > itsSCS](#)  
*our surprise computer*

### 10.228.1 Detailed Description

A red color channel.

Definition at line 195 of file ChannelsSurprise.H.

### 10.228.2 Constructor & Destructor Documentation

#### 10.228.2.1 RedChannelSurprise::RedChannelSurprise ([ModelManager](#) & mgr)

Construct with standard params.

Definition at line 184 of file ChannelsSurprise.C.

References [ModelComponent::addSubComponent\(\)](#).

#### 10.228.2.2 RedChannelSurprise::~~RedChannelSurprise () [virtual]

Destructor.

Definition at line 190 of file ChannelsSurprise.C.

### 10.228.3 Member Function Documentation

#### 10.228.3.1 [Image< float > RedChannelSurprise::computeSubmap](#) (const [uint index](#)) const [virtual]

*computeSubmap()* is special and will return surprise

Reimplemented from [SingleChannel](#).

Definition at line 193 of file ChannelsSurprise.C.

References [uint](#).

## 10.228.4 Member Data Documentation

### 10.228.4.1 [SharedPtr<SingleChannelSurprise>](#) [RedChannelSurprise::itsSCS](#) [protected]

our surprise computer

Definition at line 208 of file ChannelsSurprise.H.

The documentation for this class was generated from the following files:

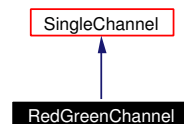
- [ChannelsSurprise.H](#)
- ChannelsSurprise.C

## 10.229 RedGreenChannel Class Reference

A red-green double-opponent color channel.

```
#include <Channels.H>
```

Inheritance diagram for RedGreenChannel: Collaboration diagram for RedGreenChannel:



### Public Member Functions

- [RedGreenChannel](#) ([ModelManager](#) &mgr)

*Construct with standard params.*

- virtual [~RedGreenChannel](#) ()

*Destructor.*

### 10.229.1 Detailed Description

A red-green double-opponent color channel.

Definition at line 113 of file Channels.H.

### 10.229.2 Constructor & Destructor Documentation

#### 10.229.2.1 RedGreenChannel::RedGreenChannel ([ModelManager](#) & mgr)

Construct with standard params.

Definition at line 152 of file Channels.C.

References [makeShared\(\)](#), [RG](#), and [ModelParam< bool >::setVal\(\)](#).

#### 10.229.2.2 RedGreenChannel::~~RedGreenChannel () [virtual]

Destructor.

Definition at line 159 of file Channels.C.

The documentation for this class was generated from the following files:

- [Channels.H](#)
- [Channels.C](#)

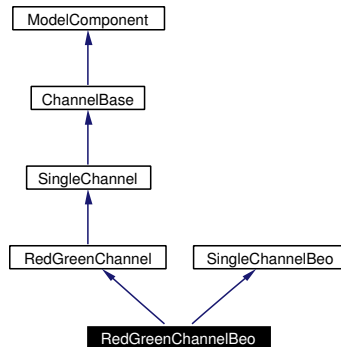


## 10.230 RedGreenChannelBeo Class Reference

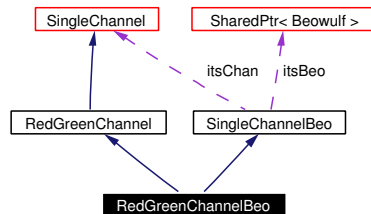
Beowulf version of [RedGreenChannel](#).

```
#include <ChannelsBeo.H>
```

Inheritance diagram for RedGreenChannelBeo:



Collaboration diagram for RedGreenChannelBeo:



### Public Member Functions

- **RedGreenChannelBeo** ([ModelManager](#) &mgr, const [SharedPtr< Beowulf >](#) &beow)
- virtual bool [outputAvailable](#) () const  
*Query whether the channel is ready to give yield valid output.*

### Protected Member Functions

- virtual void [doInput](#) (const [Image< PixRGB< float > >](#) \*, const [Image< float >](#) \*bwimg, const double t, const [Image< byte >](#) &clipMask)

*SingleChannel* requires only luminance input.

- virtual void [start1](#) ()

### 10.230.1 Detailed Description

[Beowulf](#) version of [RedGreenChannel](#).

Definition at line 67 of file ChannelsBeo.H.

### 10.230.2 Member Function Documentation

**10.230.2.1 void RedGreenChannelBeo::doInput (const [Image](#)< [PixRGB](#)< float > > \*, const [Image](#)< float > \* *bwimg*, const double *t*, const [Image](#)< byte > & *clipMask*)** [protected, virtual]

[SingleChannel](#) requires only luminance input.

Reimplemented from [SingleChannel](#).

Definition at line 86 of file ChannelsBeo.C.

References [SingleChannelBeo::sendInput\(\)](#).

**10.230.2.2 bool RedGreenChannelBeo::outputAvailable () const** [virtual]

Query whether the channel is ready to give yield valid output.

In some cases (e.g. with motion channels), it may take several input frames before the channel is fully initialized.

Reimplemented from [SingleChannel](#).

Definition at line 79 of file ChannelsBeo.C.

References [SingleChannel::outputAvailable\(\)](#).

**10.230.2.3 void RedGreenChannelBeo::start1 ()** [protected, virtual]

If you overload this, make sure you call [SingleChannel::start1\(\)](#) at the beginning of your overload

Reimplemented from [SingleChannel](#).

Definition at line 90 of file ChannelsBeo.C.

References [SingleChannelBeo::requestBeowulfNode\(\)](#), and [SingleChannel::start1\(\)](#).

The documentation for this class was generated from the following files:

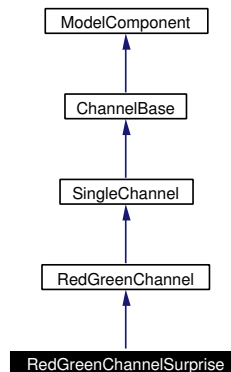
- [ChannelsBeo.H](#)
- [ChannelsBeo.C](#)

## 10.231 RedGreenChannelSurprise Class Reference

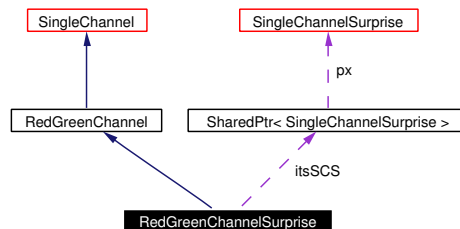
A red-green double-opponent color channel.

```
#include <ChannelsSurprise.H>
```

Inheritance diagram for RedGreenChannelSurprise:



Collaboration diagram for RedGreenChannelSurprise:



### Public Member Functions

- [RedGreenChannelSurprise](#) ([ModelManager](#) &mgr)  
*Construct with standard params.*
- virtual [~RedGreenChannelSurprise](#) ()  
*Destructor.*
- virtual [Image](#)< float > [computeSubmap](#) (const [uint](#) index) const  
*[computeSubmap\(\)](#) is special and will return surprise*

## Protected Attributes

- [SharedPtr< SingleChannelSurprise >](#) itsSCS  
*our surprise computer*

### 10.231.1 Detailed Description

A red-green double-opponent color channel.

Definition at line 87 of file ChannelsSurprise.H.

### 10.231.2 Constructor & Destructor Documentation

#### 10.231.2.1 RedGreenChannelSurprise::RedGreenChannelSurprise ([ModelManager](#) & mgr)

Construct with standard params.

Definition at line 94 of file ChannelsSurprise.C.

References [ModelComponent::addSubComponent\(\)](#).

#### 10.231.2.2 RedGreenChannelSurprise::~~RedGreenChannelSurprise () [virtual]

Destructor.

Definition at line 100 of file ChannelsSurprise.C.

### 10.231.3 Member Function Documentation

#### 10.231.3.1 [Image< float >](#) RedGreenChannelSurprise::computeSubmap (const [uint index](#)) const [virtual]

[computeSubmap\(\)](#) is special and will return surprise

Reimplemented from [SingleChannel](#).

Definition at line 103 of file ChannelsSurprise.C.

References [uint](#).

## 10.231.4 Member Data Documentation

### 10.231.4.1 [SharedPtr<SingleChannelSurprise>](#) [RedGreenChannelSurprise::itsSCS](#) [protected]

our surprise computer

Definition at line 100 of file ChannelsSurprise.H.

The documentation for this class was generated from the following files:

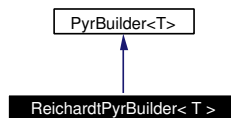
- [ChannelsSurprise.H](#)
- ChannelsSurprise.C

## 10.232 ReichardtPyrBuilder< T > Class Template Reference

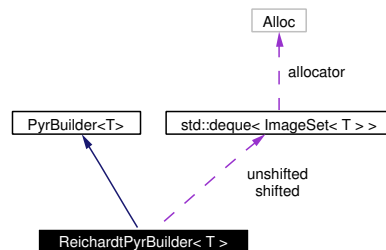
This class implements Reichardt motion detections.

```
#include <PyrBuilder.H>
```

Inheritance diagram for ReichardtPyrBuilder< T >:



Collaboration diagram for ReichardtPyrBuilder< T >:



### Public Member Functions

- [ReichardtPyrBuilder](#) (const float dx, const float dy, const [PyramidType](#) typ, const float gabor\_theta=0.0F, const float intens=10.0F)  
*build with depth, dx, dy and type, but no input image yet*
- virtual [ReichardtPyrBuilder< T > \\* clone \(\)](#) const  
*cloning constructor*
- virtual [ImageSet< T > build](#) (const [Image< T >](#) &image, const int depth)  
*Builds a Reichardt pyramid based on a image.*
- virtual void [reset \(\)](#)  
*reset the stored queues - overrides [PyrBuilder<T>::reset\(\)](#)*

### 10.232.1 Detailed Description

**template<class T> class ReichardtPyrBuilder< T >**

This class implements Reichardt motion detections.

The class uses the filter results of pyramids of other types to implement Reichardt motion detection. The pyramids of the previous time step are stored and used with the pyramid of this time step and a version of the current pyramid, in which each level is shifted by the same amount *dx* and *dy*. The results are stored in the Reichardt pyramid and represent motion at different speeds and spatial resolutions.

Definition at line 253 of file PyrBuilder.H.

### 10.232.2 Constructor & Destructor Documentation

**10.232.2.1** **template<class T> ReichardtPyrBuilder< T**  
**>::ReichardtPyrBuilder** (const float *dx*, const float *dy*, const  
**PyramidType** *typ*, const float *gabor\_theta* = 0.0F, const float *intens* =  
 10.0F)

build with depth, *dx*, *dy* and type, but no input image yet

**Parameters:**

***dx*** the number of pixels to shift into the x-direction (can be fractional value)

***dy*** the number of pixels to shift into the y-direction

***typ*** the type of pyramid to use for the filtering

***gabor\_theta*** orientation of a gabor filter (if *typ* is GaborX)

***intens*** used for gabor filter ...

Definition at line 220 of file PyrBuilder.C.

### 10.232.3 Member Function Documentation

**10.232.3.1** **template<class T> ImageSet< T > ReichardtPyrBuilder< T**  
**>::build** (const Image< T > & *image*, const int *depth*) [virtual]

Builds a Reichardt pyramid based on a image.

Implements [PyrBuilder< T >](#).

Definition at line 230 of file PyrBuilder.C.

References [buildPyrGeneric\(\)](#), [image](#), and [shiftImage\(\)](#).



**10.232.3.2** `template<class T> ReichardtPyrBuilder< T > *  
ReichardtPyrBuilder< T >::clone () const [virtual]`

cloning constructor

Implements [PyrBuilder< T >](#).

Definition at line 277 of file [PyrBuilder.C](#).

**10.232.3.3** `template<class T> void ReichardtPyrBuilder< T >::reset ()  
[virtual]`

reset the stored queues - overrides [PyrBuilder<T>::reset\(\)](#)

Reimplemented from [PyrBuilder< T >](#).

Definition at line 282 of file [PyrBuilder.C](#).

The documentation for this class was generated from the following files:

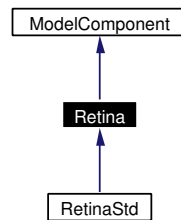
- [PyrBuilder.H](#)
- [PyrBuilder.C](#)

## 10.233 Retina Class Reference

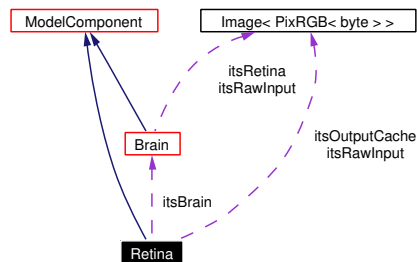
The retina base class.

```
#include <Retina.H>
```

Inheritance diagram for Retina:



Collaboration diagram for Retina:



### Public Member Functions

- `Retina` (`ModelManager` &mgr, const `std::string` &descrName="Retina", const `std::string` &tagName="Retina")

*Constructor.*

- virtual void `setBrain` (`Brain` \*brain)

*Set our brain.*

- virtual `~Retina` ()

*Destructor.*

### Access functions

- virtual void [input](#) (const [Image](#)< [PixRGB](#)< [byte](#) > > &in)  
*Input a new frame.*
- virtual bool [hasInput](#) () const  
*Have we received an input yet?*
- virtual [Image](#)< [PixRGB](#)< [byte](#) > > [getRawInput](#) () const  
*Get latest raw input frame.*
- virtual [Image](#)< [PixRGB](#)< [byte](#) > > [getOutput](#) ()  
*Get the output of the retina.*
- virtual [Rectangle](#) [getRawInputRectangle](#) (const int border=0) const  
*Get a rectangle delineating the raw input image area.*
- virtual [Point2D](#) [rawToRetinal](#) (const [Point2D](#) &rawpos) const  
*Translate a raw eye position from stimulus coordinates to retinal.*
- virtual [Point2D](#) [retinalToRaw](#) (const [Point2D](#) &retpos) const  
*Translate a retinal eye position back into raw stimulus coordinates.*

## Protected Member Functions

- virtual [Image](#)< [PixRGB](#)< [byte](#) > > [doGetOutput](#) ()=0  
*Compute a new output.*

## Protected Attributes

- [Brain](#) \* [itsBrain](#)  
*keep track of our [Brain](#)*
- [Image](#)< [PixRGB](#)< [byte](#) > > [itsRawInput](#)  
*last raw input image received*
- [Image](#)< [PixRGB](#)< [byte](#) > > [itsOutputCache](#)  
*last output computed*

### 10.233.1 Detailed Description

The retina base class.

The Retina is the entry point to the [Brain](#) and [VisualCortex](#), and this is a base class that mostly specifies the interface. Every input image to be processed first passes through the Retina (which is a subcomponent of [Brain](#)). The Retina may transform the input in various ways, e.g., apply a foveation filter, embed the input into a larger background framing image, or shift the image to current eye position.

Definition at line 57 of file Retina.H.

## 10.233.2 Constructor & Destructor Documentation

### 10.233.2.1 Retina::Retina ([ModelManager](#) & mgr, const [std::string](#) & descrName = "Retina", const [std::string](#) & tagName = "Retina")

Constructor.

See [ModelComponent.H](#) for details

Definition at line 52 of file Retina.C.

### 10.233.2.2 Retina::~~Retina () [virtual]

Destructor.

Definition at line 63 of file Retina.C.

## 10.233.3 Member Function Documentation

### 10.233.3.1 virtual [Image](#)< [PixRGB](#)<[byte](#)> > Retina::doGetOutput () [protected, pure virtual]

Compute a new output.

Derived classes should overload this, which is called by [getOutput\(\)](#)

Implemented in [RetinaStd](#).

Referenced by [getOutput\(\)](#).

### 10.233.3.2 [Image](#)< [PixRGB](#)<[byte](#)> > Retina::getOutput () [virtual]

Get the output of the retina.

The retina may apply a number of transforms to the input such as to yield the output, depending on [ModelParam](#) parameter values. This includes embedding the input within a larger background image, shifting it to current eye position, etc. Derived classes should not overload this [getOutput\(\)](#) method, as it is a caching wrapper around [doGetOutput\(\)](#) which should be overloaded.

Definition at line 79 of file Retina.C.

References `doGetOutput()`, and `Image< PixRGB< byte > >::initialized()`.

#### 10.233.3.3 `Image< PixRGB< byte > > Retina::getRawInput () const` `[virtual]`

Get latest raw input frame.

The base implementation just returns `itsRawInput`

Definition at line 75 of file Retina.C.

#### 10.233.3.4 `Rectangle Retina::getRawInputRectangle (const int border = 0)` `const [virtual]`

Get a rectangle delineating the raw input image area.

As the raw input may be embedded into a background, shifted, cropped by a field of view, etc., this function allows users to recover the boundaries of the original frames. Note that the rectangle will be possibly cropped to fit within the dims of the Retina's output. A border may be specified that will be used to shrink (if positive) or enlarge (if negative) the returned rectangle (and will be applied before the rectangle is cropped to dims). The default implementation in the base class is to just return a rectangle at (0,0) with dims of the raw input.

Reimplemented in [RetinaStd](#).

Definition at line 89 of file Retina.C.

References `Rectangle::fitToDims()`, `Image< PixRGB< byte > >::getDims()`, `Image< PixRGB< byte > >::getHeight()`, and `Image< PixRGB< byte > >::getWidth()`.

#### 10.233.3.5 `bool Retina::hasInput () const` `[virtual]`

Have we received an input yet?

The base implementation just checks whether `itsRawInput` is initialized

Definition at line 71 of file Retina.C.

References `Image< PixRGB< byte > >::initialized()`.

#### 10.233.3.6 `void Retina::input (const Image< PixRGB< byte > > & in)` `[virtual]`

Input a new frame.

The base implementation just keeps a copy of the input image in `itsRawInput`.

**Parameters:**

*in* the raw input image (e.g., from an [InputFrameSeries](#)).

Definition at line 67 of file Retina.C.

References `Image< PixRGB< byte > >::freeMem()`.

**10.233.3.7 [Point2D](#) Retina::rawToRetinal (const [Point2D](#) & rawpos) const [virtual]**

Translate a raw eye position from stimulus coordinates to retinal.

The raw eye position should be given in the coordinates of the raw input frame (before shifting, embedding, etc). The returned position is in retinal coordinates, taking into account possible shifting, framing, fov'ing, etc. Default implementation in the base class is a no-op.

Reimplemented in [RetinaStd](#).

Definition at line 99 of file Retina.C.

**10.233.3.8 [Point2D](#) Retina::retinalToRaw (const [Point2D](#) & retpos) const [virtual]**

Translate a retinal eye position back into raw stimulus coordinates.

The returned raw eye position is in the coordinates of the raw input frame (before shifting, embedding, etc). The given retinal position is in retinal coordinates, taking into account possible shifting, framing, fov'ing, etc. Default implementation in the base class is a no-op.

Reimplemented in [RetinaStd](#).

Definition at line 103 of file Retina.C.

**10.233.3.9 void Retina::setBrain ([Brain](#) \* brain) [virtual]**

Set our brain.

Some viewings may require a [Brain](#). Since in the base class we don't have a [Brain](#) data member, the base class implementation of this function is a no-op. Derived classes should overload this accordingly if they need a [Brain](#).

Definition at line 59 of file Retina.C.

## 10.233.4 Member Data Documentation

### 10.233.4.1 [Brain\\*](#) [Retina::itsBrain](#) [protected]

keep track of our [Brain](#)

Definition at line 139 of file Retina.H.

### 10.233.4.2 [Image](#)< [PixRGB](#)<[byte](#)> > [Retina::itsOutputCache](#) [protected]

last output computed

Definition at line 141 of file Retina.H.

### 10.233.4.3 [Image](#)< [PixRGB](#)<[byte](#)> > [Retina::itsRawInput](#) [protected]

last raw input image received

Definition at line 140 of file Retina.H.

The documentation for this class was generated from the following files:

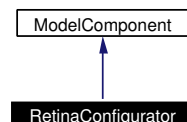
- [Retina.H](#)
- [Retina.C](#)

## 10.234 RetinaConfigurator Class Reference

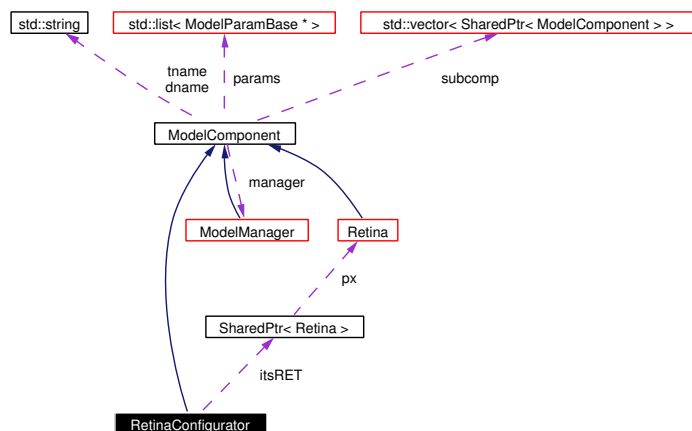
[Retina](#) configurator.

```
#include <Retina.H>
```

Inheritance diagram for RetinaConfigurator:



Collaboration diagram for RetinaConfigurator:



### Public Member Functions

- [RetinaConfigurator](#) ([ModelManager](#) &mgr, const [std::string](#) &descr-Name="Retina Configurator", const [std::string](#) &tagName="Retina-Configurator")

*Constructor.*

- virtual [~RetinaConfigurator](#) ()

*destructor*

- virtual void [exportOptions](#) (const int whatToExport, const bool recurse=true)

*Export some of our [ModelParam](#) parameters as command-line-options.*



- `SharedPtr< Retina > getRET () const`

*Get the chosen Retina.*

## Protected Member Functions

- virtual bool `setModelParamString` (const `std::string` &name, const `std::string` &value, const bool recurse=false, const bool warn=true)

*Intercept people changing our ModelParam.*

## Protected Attributes

- `ModelParam< std::string > itsType`

*type of retina*

### 10.234.1 Detailed Description

`Retina` configurator.

This will export the `--retina-type=XX` command-line option and will instantiate a `Retina` of the desired type as the option gets assigned a value. As this happens, new options may become available in the command-line. To see them, use `--help AFTER` you have chosen the type to use. The current `Retina` may be retrieved using `getRET()`.

Definition at line 153 of file `Retina.H`.

### 10.234.2 Constructor & Destructor Documentation

- #### 10.234.2.1 RetinaConfigurator::RetinaConfigurator (`ModelManager` & mgr, const `std::string` & descrName = "Retina Configurator", const `std::string` & tagName = "RetinaConfigurator")

Constructor.

Definition at line 113 of file `Retina.C`.

References `OPTEXP_NONE`.

### 10.234.2.2 RetinaConfigurator::~~RetinaConfigurator () [virtual]

destructor

Definition at line 122 of file Retina.C.

## 10.234.3 Member Function Documentation

### 10.234.3.1 void RetinaConfigurator::exportOptions (const int *whatToExport*, const bool *recurse* = true) [virtual]

Export some of our [ModelParam](#) parameters as command-line-options.

See the base function in [ModelComponent.H](#) for info. Actually, here we just keep a copy of *whatToExport* and we will call [exportOptions\(\)](#) on our SM once we know its type and instantiate it.

Reimplemented from [ModelComponent](#).

Definition at line 126 of file Retina.C.

References [ModelComponent::exportOptions\(\)](#), [OPTEXP\\_CORE](#), and [ModelManager::requestOption\(\)](#).

### 10.234.3.2 SharedPtr< Retina > RetinaConfigurator::getRET () const

Get the chosen [Retina](#).

You should call this during [start\(\)](#) of the [ModelComponent](#) that needs the [Retina](#). CAUTION: this may be a [SharedPtr\(NULL\)](#) if no [Retina](#) type has been chosen.

Definition at line 142 of file Retina.C.

### 10.234.3.3 bool RetinaConfigurator::setModelParamString (const std::string & *name*, const std::string & *value*, const bool *recurse* = false, const bool *warn* = true) [protected, virtual]

Intercept people changing our [ModelParam](#).

See [ModelComponent.H](#); as parsing the command-line or reading a config file sets our name, we'll also here instantiate a retina of the proper type (and export its options)

Reimplemented from [ModelComponent](#).

Definition at line 146 of file Retina.C.

References [ModelComponent::addSubComponent\(\)](#), [SharedPtr< Retina >::get\(\)](#), [ModelComponent::removeSubComponent\(\)](#), [SharedPtr< Retina >::reset\(\)](#), and [ModelComponent::setModelParamString\(\)](#).

## 10.234.4 Member Data Documentation

### 10.234.4.1 [ModelParam<std::string>](#) [RetinaConfigurator::itsType](#) [protected]

type of retina

Definition at line 179 of file Retina.H.

The documentation for this class was generated from the following files:

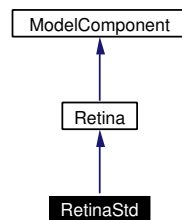
- [Retina.H](#)
- [Retina.C](#)

## 10.235 RetinaStd Class Reference

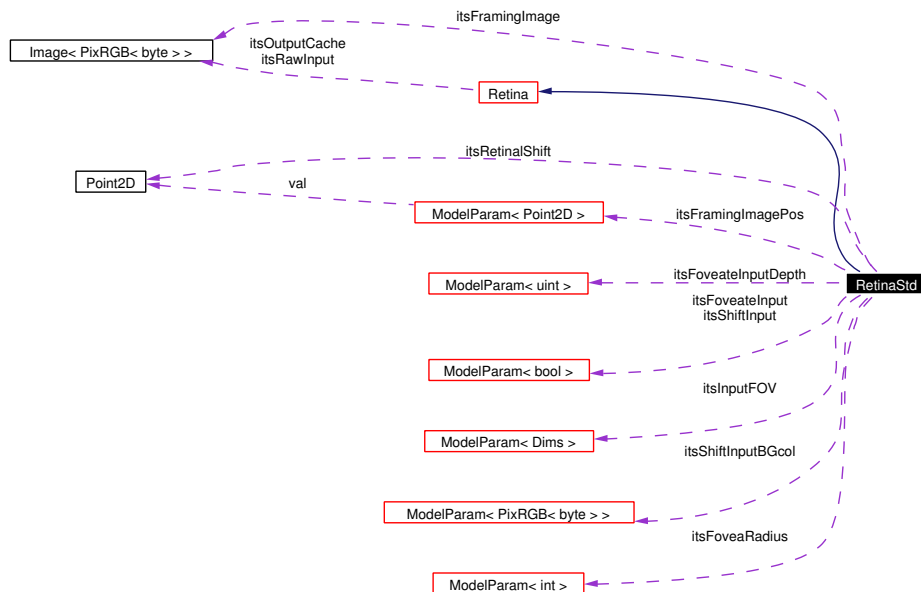
Standard [Retina](#) implementation.

```
#include <Retina.H>
```

Inheritance diagram for RetinaStd:



Collaboration diagram for RetinaStd:



### Public Member Functions

- [RetinaStd](#) ([ModelManager](#) &mgr, const [std::string](#) &descrName="RetinaStd", const [std::string](#) &tagName="RetinaStd")

*Constructor.*

- virtual void [exportOptions](#) (const int whatToExport, const bool recurse=true)  
*Export some of our [ModelParam](#) parameters as command-line-options.*
- virtual [~RetinaStd](#) ()

*Destructor.*

### Access functions

- virtual [Rectangle](#) [getRawInputRectangle](#) (const int border=0) const  
*Get a rectangle delineating the raw input image area.*
- template<class T> [Image](#)< T > [frameImage](#) (const [Image](#)< T > &img, const [Point2D](#) &pos) const  
*Frame an image.*
- template<class T> [Image](#)< [PixRGB](#)< T > > [frameImage](#) (const [Image](#)< [PixRGB](#)< T > > &img, const [Point2D](#) &pos) const
- [Point2D](#) [getFramingOffset](#) () const  
*Return current framing offset.*
- virtual [Point2D](#) [rawToRetinal](#) (const [Point2D](#) &rawpos) const  
*Translate a raw eye position from stimulus coordinates to retinal.*
- virtual [Point2D](#) [retinalToRaw](#) (const [Point2D](#) &retpos) const  
*Translate a retinal eye position back into raw stimulus coordinates.*

### Protected Member Functions

- virtual [Image](#)< [PixRGB](#)< byte > > [doGetOutput](#) ()  
*Compute a new output.*
- virtual void [start2](#) ()  
*get started*

### Protected Attributes

- [ModelParam](#)< std::string > [itsFramingImageName](#)  
*Image file to use for input framing.*

- [ModelParam](#)< [Point2D](#) > [itsFramingImagePos](#)  
*Offset to apply for input framing.*
- [ModelParam](#)< bool > [itsFoveateInput](#)  
*foveate the input?*
- [ModelParam](#)< uint > [itsFoveateInputDepth](#)  
*Pyramid depth to use when foveating the input if itsFoveateInput true.*
- [ModelParam](#)< bool > [itsShiftInput](#)  
*shift input to eye position?*
- [ModelParam](#)< [PixRGB](#)< byte > > [itsShiftInputBGcol](#)  
*background color to use when shifting input to eye position*
- [ModelParam](#)< [Dims](#) > [itsInputFOV](#)  
*crop input to a given FOV?*
- [ModelParam](#)< int > [itsFoveaRadius](#)  
*radius of the Fovea, in pixels*

### 10.235.1 Detailed Description

Standard [Retina](#) implementation.

Can do foveation, shifting, framing.

Definition at line 199 of file [Retina.H](#).

### 10.235.2 Constructor & Destructor Documentation

#### 10.235.2.1 [RetinaStd::RetinaStd](#) ([ModelManager](#) & mgr, const [std::string](#) & descrName = "RetinaStd", const [std::string](#) & tagName = "RetinaStd")

Constructor.

See [ModelComponent.H](#) for details

Definition at line 190 of file [Retina.C](#).

**10.235.2.2 RetinaStd::~~RetinaStd () [virtual]**

Destructor.

Definition at line 237 of file Retina.C.

**10.235.3 Member Function Documentation****10.235.3.1 Image< PixRGB< byte > > RetinaStd::doGetOutput () [protected, virtual]**

Compute a new output.

Implements [Retina](#).

Definition at line 241 of file Retina.C.

References [buildPyrGaussian\(\)](#), [crop\(\)](#), [Image< T >::drawDisk\(\)](#), [eye\(\)](#), [Image< T >::getDims\(\)](#), [Image< T >::getHeight\(\)](#), [Brain::getSC\(\)](#), [ModelParam< int >::getVal\(\)](#), [ModelParam< uint >::getVal\(\)](#), [ModelParam< Dims >::getVal\(\)](#), [ModelParam< PixRGB< byte > >::getVal\(\)](#), [ModelParam< bool >::getVal\(\)](#), [ModelParam< Point2D >::getVal\(\)](#), [Image< T >::getWidth\(\)](#), [Dims::h\(\)](#), [Brain::hasModule\(\)](#), [Point2D::i](#), [Image< PixRGB< byte > >::initialized\(\)](#), [Dims::isEmpty\(\)](#), [itsFoveateInput](#), [itsFoveateInputDepth](#), [itsFramingImagePos](#), [itsInputFOV](#), [itsShiftInput](#), [itsShiftInputBGcol](#), [Point2D::j](#), [Image< T >::paste\(\)](#), [shiftClean\(\)](#), [Dims::w\(\)](#), and [ZE-ROS](#).

**10.235.3.2 void RetinaStd::exportOptions (const int *whatToExport*, const bool *recurse* = true) [virtual]**

Export some of our [ModelParam](#) parameters as command-line-options.

See the base function in [ModelComponent.H](#) for info.

Reimplemented from [ModelComponent](#).

Definition at line 205 of file Retina.C.

References [ModelComponent::exportOptions\(\)](#), [itsFoveateInput](#), [itsFoveateInputDepth](#), [itsFramingImageName](#), [itsFramingImagePos](#), [itsInputFOV](#), [itsShiftInput](#), [itsShiftInputBGcol](#), [OPTEXP\\_CORE](#), and [ModelManager::requestOption\(\)](#).

**10.235.3.3 template<class T> Image<T> RetinaStd::frameImage (const Image< T > & *img*, const Point2D & *pos*) const**

Frame an image.

This will embed the given image into a larger 'framing image' that may be specified

via command-line options. The goal is to add a standard border around every incoming image. The image will be embedded with its top-left corner at the position specified via the command-line. The offset is available via [getFramingOffset\(\)](#) at any time after [start\(\)](#). This function is provided in the public section so that users can frame various images themselves, but note that it is also called internally by [getOutput\(\)](#) as soon as the relevant command-line options have non-empty values.

#### 10.235.3.4 [Point2D](#) RetinaStd::getFramingOffset () const

Return current framing offset.

This returns the coordinates of the top-left corner of the frames read from disk, once they have been 'framed' into a larger static background image using [frameImage\(\)](#).

#### 10.235.3.5 [Rectangle](#) RetinaStd::getRawInputRectangle (const int *border* = 0) const [virtual]

Get a rectangle delineating the raw input image area.

Reimplemented from [Retina](#).

Definition at line 331 of file Retina.C.

References [Rectangle::fitToDims\(\)](#), [Image< PixRGB< byte > >::getDims\(\)](#), [Image< PixRGB< byte > >::getHeight\(\)](#), [ModelParam< Point2D >::getVal\(\)](#), [Image< PixRGB< byte > >::getWidth\(\)](#), [Point2D::i](#), [itsFramingImagePos](#), and [Point2D::j](#).

#### 10.235.3.6 [Point2D](#) RetinaStd::rawToRetinal (const [Point2D](#) & *rawpos*) const [virtual]

Translate a raw eye position from stimulus coordinates to retinal.

Reimplemented from [Retina](#).

Definition at line 343 of file Retina.C.

References [ModelParam< Point2D >::getVal\(\)](#), [Point2D::i](#), [itsFramingImagePos](#), and [Point2D::j](#).

#### 10.235.3.7 [Point2D](#) RetinaStd::retinalToRaw (const [Point2D](#) & *retpos*) const [virtual]

Translate a retinal eye position back into raw stimulus coordinates.

Reimplemented from [Retina](#).

Definition at line 359 of file Retina.C.



References `ModelParam< Point2D >::getVal()`, `Point2D::i`, `itsFramingImagePos`, and `Point2D::j`.

#### 10.235.3.8 `void RetinaStd::start2 ()` [protected, virtual]

get started

Reimplemented from [ModelComponent](#).

Definition at line 224 of file `Retina.C`.

References `Image< PixRGB< byte > >::getHeight()`, `Image< PixRGB< byte > >::getWidth()`, `itsFramingImageName`, and `Raster::ReadRGB()`.

### 10.235.4 Member Data Documentation

#### 10.235.4.1 `ModelParam<int> RetinaStd::itsFoveaRadius` [protected]

radius of the Fovea, in pixels

Definition at line 276 of file `Retina.H`.

#### 10.235.4.2 `ModelParam<bool> RetinaStd::itsFoveateInput` [protected]

foveate the input?

Definition at line 261 of file `Retina.H`.

Referenced by `doGetOutput()`, and `exportOptions()`.

#### 10.235.4.3 `ModelParam<uint> RetinaStd::itsFoveateInputDepth` [protected]

Pyramid depth to use when foveating the input if `itsFoveateInput` true.

Definition at line 264 of file `Retina.H`.

Referenced by `doGetOutput()`, and `exportOptions()`.

#### 10.235.4.4 `ModelParam<std::string> RetinaStd::itsFramingImageName` [protected]

[Image](#) file to use for input framing.

Definition at line 255 of file `Retina.H`.

Referenced by `exportOptions()`, and `start2()`.

**10.235.4.5   [ModelParam<Point2D>](#)   [RetinaStd::itsFramingImagePos](#)  
[protected]**

Offset to apply for input framing.

Definition at line 258 of file Retina.H.

Referenced by [doGetOutput\(\)](#), [exportOptions\(\)](#), [getRawInputRectangle\(\)](#), [rawToRetinal\(\)](#), and [retinalToRaw\(\)](#).

**10.235.4.6   [ModelParam<Dims>](#)   [RetinaStd::itsInputFOV](#)   [protected]**

crop input to a given FOV?

Definition at line 273 of file Retina.H.

Referenced by [doGetOutput\(\)](#), and [exportOptions\(\)](#).

**10.235.4.7   [ModelParam<bool>](#)   [RetinaStd::itsShiftInput](#)   [protected]**

shift input to eye position?

Definition at line 267 of file Retina.H.

Referenced by [doGetOutput\(\)](#), and [exportOptions\(\)](#).

**10.235.4.8   [ModelParam< \[PixRGB<byte>\]\(#\) >](#)   [RetinaStd::itsShiftInputBGcol](#)  
[protected]**

background color to use when shifting input to eye position

Definition at line 270 of file Retina.H.

Referenced by [doGetOutput\(\)](#), and [exportOptions\(\)](#).

The documentation for this class was generated from the following files:

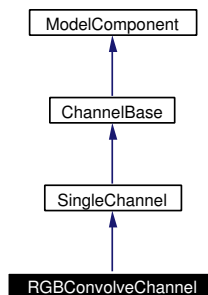
- [Retina.H](#)
- [Retina.C](#)

## 10.236 RGBConvolveChannel Class Reference

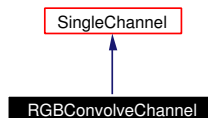
A generic convolution channel using color filters.

```
#include <Channels.H>
```

Inheritance diagram for RGBConvolveChannel:



Collaboration diagram for RGBConvolveChannel:



### Public Member Functions

- [RGBConvolveChannel](#) ([ModelManager](#) &mgr)  
*Construct with standard params. Will need to set the 3 filters.*
- void [setFilters](#) (const [Image](#)< float > &rfilter, const [Image](#)< float > &gfilter, const [Image](#)< float > &bfilter)  
*Set the 3 filters to use for convolutions.*
- virtual [~RGBConvolveChannel](#) ()  
*Destructor.*

## Protected Member Functions

- virtual void `doInput` (const `Image< PixRGB< float > > *`, const `Image< float > *bwimg`, const double `t`, const `Image< byte > &clipMask`)  
*SingleChannel requires only luminance input.*
- virtual `ImageSet< float > computePyramid` (const `Image< PixRGB< float > > &img`)  
*compute from a color image*

### 10.236.1 Detailed Description

A generic convolution channel using color filters.

Definition at line 87 of file Channels.H.

### 10.236.2 Constructor & Destructor Documentation

#### 10.236.2.1 `RGBConvolveChannel::RGBConvolveChannel (ModelManager & mgr)`

Construct with standard params. Will need to set the 3 filters.

Definition at line 109 of file Channels.C.

References `RGBCONVOLVE`, and `ModelParam< bool >::setVal()`.

#### 10.236.2.2 `RGBConvolveChannel::~~RGBConvolveChannel ()` [virtual]

Destructor.

Definition at line 126 of file Channels.C.

### 10.236.3 Member Function Documentation

#### 10.236.3.1 `ImageSet< float > RGBConvolveChannel::computePyramid (const Image< PixRGB< float > > & img)` [protected, virtual]

compute from a color image

Definition at line 138 of file Channels.C.

References `SharedPtr< T >::dynCastFrom()`, `ModelParam< LevelSpec >::getVal()`, and `LevelSpec::maxDepth()`.

Referenced by `doInput()`.

**10.236.3.2** `void RGBConvolveChannel::doInput (const Image< PixRGB< float > > *, const Image< float > * bwimg, const double t, const Image< byte > & clipMask)` [protected, virtual]

[SingleChannel](#) requires only luminance input.

Reimplemented from [SingleChannel](#).

Definition at line 128 of file `Channels.C`.

References `ASSERT`, `computePyramid()`, `SingleChannel::setClipPyramid()`, and `SingleChannel::storePyramid()`.

**10.236.3.3** `void RGBConvolveChannel::setFilters (const Image< float > & rfilter, const Image< float > & gfilter, const Image< float > & bfilter)`

Set the 3 filters to use for convolutions.

Definition at line 117 of file `Channels.C`.

References `makeShared()`, and `SingleChannel::setPyramid()`.

The documentation for this class was generated from the following files:

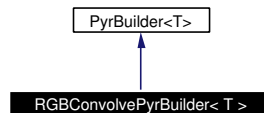
- [Channels.H](#)
- [Channels.C](#)

## 10.237 RGBConvolvePyrBuilder< T > Class Template Reference

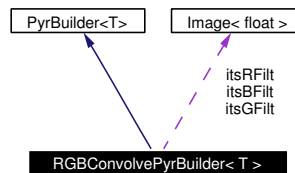
Builds pyramids based on arbitrary color filters.

```
#include <PyrBuilder.H>
```

Inheritance diagram for RGBConvolvePyrBuilder< T >:



Collaboration diagram for RGBConvolvePyrBuilder< T >:



### Public Member Functions

- **RGBConvolvePyrBuilder** (const **Image**< float > &rfilt, const **Image**< float > &gfilt, const **Image**< float > &bfilt)  
*Constructor.*
- virtual **RGBConvolvePyrBuilder**< T > \* **clone** () const  
*Cloning constructor:.*
- virtual **ImageSet**< T > **build** (const **Image**< **PixRGB**< T > > &img, const int depth)  
*Builds a pyramid based on a img.*
- virtual **ImageSet**< T > **build** (const **Image**< T > &img, const int depth)  
*Don't use this one!*
- **ImageSet**< **PixRGB**< T > > **build2** (const **Image**< **PixRGB**< T > > &img, const int depth)

### 10.237.1 Detailed Description

**template<class T> class RGBConvolvePyrBuilder< T >**

Builds pyramids based on arbitrary color filters.

Definition at line 136 of file PyrBuilder.H.

### 10.237.2 Constructor & Destructor Documentation

**10.237.2.1** **template<class T> RGBConvolvePyrBuilder< T**  
**>::RGBConvolvePyrBuilder** (const **Image**< float > & *rfilt*, const  
**Image**< float > & *gfil*, const **Image**< float > & *bfil*)

Constructor.

Definition at line 105 of file PyrBuilder.C.

### 10.237.3 Member Function Documentation

**10.237.3.1** **template<class T> ImageSet< T > RGBConvolvePyrBuilder< T**  
**>::build** (const **Image**< T > & *img*, const int *depth*) [virtual]

Don't use this one!

Implements **PyrBuilder< T >**.

Definition at line 128 of file PyrBuilder.C.

**10.237.3.2** **template<class T> ImageSet< T > RGBConvolvePyrBuilder< T**  
**>::build** (const **Image**< **PixRGB**< T > > & *img*, const int *depth*)  
 [virtual]

Builds a pyramid based on a img.

Definition at line 112 of file PyrBuilder.C.

References **buildPyrConvolve()**, and **getComponents()**.

**10.237.3.3** **template<class T> ImageSet< PixRGB< T > >**  
**RGBConvolvePyrBuilder< T >::build2** (const **Image**< **PixRGB**< T  
**> > & img**, const int *depth*)

Builds a color pyramid based on a img. Only used to display results, filters behaviour...

Definition at line 136 of file PyrBuilder.C.

References [buildPyrConvolve\(\)](#), [getComponents\(\)](#), and [makeRGB\(\)](#).

**10.237.3.4** `template<class T> RGBConvolvePyrBuilder< T > *  
RGBConvolvePyrBuilder< T >::clone () const [virtual]`

Cloning constructor:.

Implements [PyrBuilder< T >](#).

Definition at line 152 of file [PyrBuilder.C](#).

The documentation for this class was generated from the following files:

- [PyrBuilder.H](#)
- [PyrBuilder.C](#)

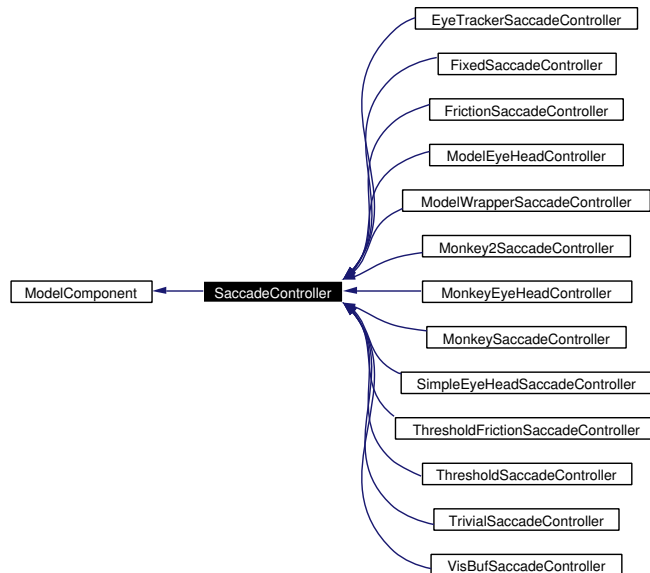


## 10.238 SaccadeController Class Reference

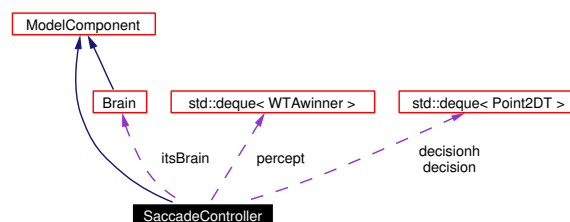
A base class to implement saccade controllers.

```
#include <SaccadeController.H>
```

Inheritance diagram for SaccadeController:



Collaboration diagram for SaccadeController:



### Public Member Functions

- [SaccadeController](#) ([ModelManager](#) &mgr, const [std::string](#) &descrName, const [std::string](#) &tagName, const int percept\_qlen, const int decision\_qlen, const int decision\_qlen\_head=0)

*Constructor.*

- virtual void [setBrain](#) ([Brain](#) \*brain)  
*Set a hook to our [Brain](#).*
- virtual [~SaccadeController](#) ()  
*Destructor (virtual to ensure that derived classes correctly destroyed).*
- virtual void [reset](#) (const bool recurse=true)  
*Reset SaccadeController.*
- virtual void [setPercept](#) (const [WTAWinner](#) &fix)  
*Receive a new percept (e.g., winner from saliency map).*
- virtual void [resetPos](#) (const [Point2DT](#) &p)  
*Reset to an externally-given position.*
- virtual [Point2D](#) [getDecision](#) (const double t)  
*Get the fixation decided by the controller.*
- virtual [Point2D](#) [getDecisionHead](#) (const double t)  
*Same as [getDecision\(\)](#) but for the head position.*
- [WTAWinner](#) [getPreviousPercept](#) (const unsigned int index=0) const  
*Get a previous percept.*
- bool [havePercepts](#) () const  
*Do we already have any percepts queued up?*
- int [getPqlen](#) () const  
*Returns length of percept queue.*
- void [resetPqlen](#) (const int len)  
*Reset length of percept queue (contents will be lost).*
- void [killPercepts](#) ()  
*Kill all previous percepts (empty the percept queue).*
- [Point2DT](#) [getPreviousDecision](#) (const unsigned int index=0) const  
*Get a previous decision.*
- [Point2DT](#) [getPreviousDecisionHead](#) (const unsigned int index=0) const  
*Same as [getPreviousDecision](#) but for the head.*

- bool `haveDecisions ()` const  
*Do we already have any decisions queued up?*
- int `getDqlen ()` const  
*Returns length of decision queue.*
- void `resetDqlen (const int len)`  
*Reset length of decision queue (contents will be lost).*
- void `killDecisions ()`  
*Kill all previous decisions (empty the decision queue).*
- bool `haveDecisionsHead ()` const  
*Do we already have any head decisions queued up?*
- int `getDqlenHead ()` const  
*Returns length of head decision queue.*
- void `resetDqlenHead (const int len)`  
*Reset length of head decision queue (contents will be lost).*
- void `killDecisionsHead ()`  
*Kill all previous decisions (empty the decision queue).*
- virtual void `evolve (const double t)=0`  
*evolve one time step (called, e.g., by `Brain::evolve()`)*
- bool `isInSaccade ()` const  
*Are we executing a saccade?*
- bool `isInSaccadeHead ()` const  
*Are we executing a head "saccade"?*
- bool `isInBlink ()` const  
*Are we executing an eye blink?*
- void `dumpQueues ()` const  
*Show contents of the queues, for debugging.*

## Protected Member Functions

- void [beginSaccade](#) ()  
*Call this when you start a saccade (will make [isInSaccade\(\)](#) true).*
- void [endSaccade](#) ()  
*Call this when you end a saccade (will make [isInSaccade\(\)](#) false).*
- void [beginBlink](#) ()  
*Call this when you start a blink (will make [isInBlink\(\)](#) true).*
- void [endBlink](#) ()  
*Call this when you end a blink (will make [isInBlink\(\)](#) false).*
- void [beginSaccadeHead](#) ()  
*Call this when you start a saccade (will make [isInSaccade\(\)](#) true).*
- void [endSaccadeHead](#) ()  
*Call this when you end a saccade (will make [isInSaccade\(\)](#) false).*
- virtual void [computeWhenNewPercept](#) ()=0  
*This method is called each time a new percept has arrived.*
- virtual void [computeWhenResetPos](#) ()=0  
*This method is called each time a [reset\(\)](#) occurs.*
- virtual [Point2D](#) [computeWhenNewDecision](#) (const double t)=0  
*This method is called each time a new decision is requested.*
- virtual [Point2D](#) [computeWhenNewDecisionHead](#) (const double t)  
*Same as [computeWhenNewDecision](#) but for the head.*

## Protected Attributes

- [Brain](#) \* [itsBrain](#)  
*hook to our [Brain](#) or NULL*

### 10.238.1 Detailed Description

A base class to implement saccade controllers.

This base class handles the basic queuing of percepts and fixations. Typically, new percepts (e.g., winner-take-all winners over the saliency map) should be passed to the saccade controller using [setPercept\(\)](#). During normal evolution of a simulation, [evolve\(\)](#) should be called at every time step, in a manner similar to that of [Brain](#), WTA and SM. In order to obtain coordinates of saccade targets computed by this class, [getDecision\(\)](#) should be called. It will return (-1, -1) if no new saccade target is decided (either because it cannot be decided, or because it is the same as the last valid returned one). So, saccading to and fixating a new target will yield one initial valid set of returned coordinates, followed by subsequent returns of (-1, -1) as long as fixation is to be held stationary. Derived classes should overload [computeWhenNewPercept\(\)](#), called each time a new percept has arrived, and [computeWhenNewDecision\(\)](#), called each time a new decision is requested. In addition, derived classes should usually overload [evolve\(\)](#), typically called at each time step during a simulation. For example, the [Brain](#) class needs a SaccadeController derivative object that implements [evolve\(\)](#), though that implementation may be a no-op. The controller may include a head movement component; this works just like the eye movement component. If no head movement is being simulated, just use a decision queue length of zero for the head. Callers can check whether a head is simulated using [haveDecisionsHead\(\)](#). See [SaccadeControllers.H](#) for examples of derived classes which actually implement saccadic control strategies.

Definition at line 73 of file SaccadeController.H.

### 10.238.2 Constructor & Destructor Documentation

**10.238.2.1 SaccadeController::SaccadeController ([ModelManager](#) & mgr, const [std::string](#) & descrName, const [std::string](#) & tagName, const int percept\_qlen, const int decision\_qlen, const int decision\_qlen\_head = 0)**

Constructor.

**Parameters:**

*percept\_qlen* length of the queue of accumulated percepts

*decision\_qlen* length of the queue of generated decisions

*decision\_qlen\_head* length of the queue of generated decisions for the head. If your controller has no head component, just use 0 here.

Definition at line 41 of file SaccadeController.C.

**10.238.2.2 SaccadeController::~~SaccadeController () [virtual]**

Destructor (virtual to ensure that derived classes correctly destroyed).

Definition at line 65 of file SaccadeController.C.

**10.238.3 Member Function Documentation****10.238.3.1 void SaccadeController::beginBlink () [protected]**

Call this when you start a blink (will make [isInBlink\(\)](#) true).

Definition at line 277 of file SaccadeController.C.

Referenced by FixedSaccadeController::blink(), ModelWrapperSaccadeController::evolve(), EyeTrackerSaccadeController::evolve(), MonkeySaccadeController::evolve(), and Monkey2SaccadeController::evolve().

**10.238.3.2 void SaccadeController::beginSaccade () [protected]**

Call this when you start a saccade (will make [isInSaccade\(\)](#) true).

Definition at line 265 of file SaccadeController.C.

Referenced by ThresholdFrictionSaccadeController::computeWhenNewDecision(), ThresholdSaccadeController::computeWhenNewDecision(), ModelWrapperSaccadeController::computeWhenNewPercept(), Monkey2SaccadeController::computeWhenNewPercept(), ThresholdFrictionSaccadeController::computeWhenResetPos(), ThresholdSaccadeController::computeWhenResetPos(), FrictionSaccadeController::computeWhenResetPos(), EyeTrackerSaccadeController::evolve(), ModelEyeHeadController::saccade(), MonkeyEyeHeadController::saccade(), and FixedSaccadeController::saccade().

**10.238.3.3 void SaccadeController::beginSaccadeHead () [protected]**

Call this when you start a saccade (will make [isInSaccade\(\)](#) true).

Definition at line 285 of file SaccadeController.C.

Referenced by ModelWrapperSaccadeController::computeWhenNewPercept(), and Monkey2SaccadeController::computeWhenNewPercept().

**10.238.3.4 virtual [Point2D](#) SaccadeController::computeWhenNewDecision (const double t) [protected, pure virtual]**

This method is called each time a new decision is requested.

The latest percept is at the front of the percept queue, and the latest previous decision at the front of the decision queue. The method should return the decision to be taken given the time passed as argument. This method should return the current decision for fixation. The wrapping [getDecision\(\)](#) call will take care of eliminating decisions that are the same as the previous ones. Derived classes must provide some implementation for this method, as it is called from within [getDecision\(\)](#).

Implemented in [TrivialSaccadeController](#), [FixedSaccadeController](#), [FrictionSaccadeController](#), [ThresholdSaccadeController](#), [ThresholdFrictionSaccadeController](#), [SimpleEyeHeadSaccadeController](#), [MonkeyEyeHeadController](#), [Monkey2SaccadeController](#), [MonkeySaccadeController](#), [EyeTrackerSaccadeController](#), [VisBufSaccadeController](#), [ModelEyeHeadController](#), and [ModelWrapperSaccadeController](#).

Referenced by [getDecision\(\)](#).

#### 10.238.3.5 **Point2D SaccadeController::computeWhenNewDecisionHead (const double t)** [protected, virtual]

Same as [computeWhenNewDecision](#) but for the head.

A default no-op implementation is provided in the base class, so if your derived class has no head, you don't need to implement this.

Reimplemented in [SimpleEyeHeadSaccadeController](#), [Monkey2SaccadeController](#), and [ModelWrapperSaccadeController](#).

Definition at line 181 of file [SaccadeController.C](#).

Referenced by [getDecisionHead\(\)](#).

#### 10.238.3.6 **virtual void SaccadeController::computeWhenNewPercept ()** [protected, pure virtual]

This method is called each time a new percept has arrived.

The latest percept is at the front of the percept queue. Derived classes must provide some implementation for this method, as it is called from within [setPercept\(\)](#).

Implemented in [TrivialSaccadeController](#), [FixedSaccadeController](#), [FrictionSaccadeController](#), [ThresholdSaccadeController](#), [ThresholdFrictionSaccadeController](#), [SimpleEyeHeadSaccadeController](#), [MonkeyEyeHeadController](#), [Monkey2SaccadeController](#), [MonkeySaccadeController](#), [EyeTrackerSaccadeController](#), [VisBufSaccadeController](#), [ModelEyeHeadController](#), and [ModelWrapperSaccadeController](#).

Referenced by [setPercept\(\)](#).

### 10.238.3.7 **virtual void SaccadeController::computeWhenResetPos ()** [protected, pure virtual]

This method is called each time a [reset\(\)](#) occurs.

The reset location/time is at the front of the percept queue. Derived classes must provide some implementation for this method, as it is called from within [reset\(\)](#).

Implemented in [TrivialSaccadeController](#), [FixedSaccadeController](#), [FrictionSaccadeController](#), [ThresholdSaccadeController](#), [ThresholdFrictionSaccadeController](#), [SimpleEyeHeadSaccadeController](#), [MonkeyEyeHeadController](#), [Monkey2SaccadeController](#), [MonkeySaccadeController](#), [EyeTrackerSaccadeController](#), [VisBufSaccadeController](#), [ModelEyeHeadController](#), and [ModelWrapperSaccadeController](#).

Referenced by [resetPos\(\)](#).

### 10.238.3.8 **void SaccadeController::dumpQueues () const**

Show contents of the queues, for debugging.

Definition at line 161 of file [SaccadeController.C](#).

References [WTAwinner::av](#), [Point2D::i](#), [Point2D::j](#), [Point2DT::p](#), [WTAwinner::p](#), [WTAwinner::sv](#), [Point2DT::t](#), and [WTAwinner::t](#).

### 10.238.3.9 **void SaccadeController::endBlink ()** [protected]

Call this when you end a blink (will make [isInBlink\(\)](#) false).

Definition at line 281 of file [SaccadeController.C](#).

Referenced by [FixedSaccadeController::blink\(\)](#), [ModelWrapperSaccadeController::evolve\(\)](#), [EyeTrackerSaccadeController::evolve\(\)](#), [MonkeySaccadeController::evolve\(\)](#), and [Monkey2SaccadeController::evolve\(\)](#).

### 10.238.3.10 **void SaccadeController::endSaccade ()** [protected]

Call this when you end a saccade (will make [isInSaccade\(\)](#) false).

Definition at line 269 of file [SaccadeController.C](#).

Referenced by [MonkeySaccadeController::computeWhenNewDecision\(\)](#), [ModelEyeHeadController::computeWhenResetPos\(\)](#), [MonkeyEyeHeadController::computeWhenResetPos\(\)](#), [ModelWrapperSaccadeController::evolve\(\)](#), [EyeTrackerSaccadeController::evolve\(\)](#), [Monkey2SaccadeController::evolve\(\)](#), [ThresholdFrictionSaccadeController::evolve\(\)](#), [ThresholdSaccadeController::evolve\(\)](#), [FrictionSaccadeController::evolve\(\)](#), and [FixedSaccadeController::saccade\(\)](#).



**10.238.3.11 void SaccadeController::endSaccadeHead ()** [protected]

Call this when you end a saccade (will make [isInSaccade\(\)](#) false).

Definition at line 289 of file SaccadeController.C.

Referenced by [ModelWrapperSaccadeController::evolve\(\)](#), and [Monkey2SaccadeController::evolve\(\)](#).

**10.238.3.12 virtual void SaccadeController::evolve (const double t)** [pure virtual]

evolve one time step (called, e.g., by [Brain::evolve\(\)](#))

This method should be invoked by external callers to allow the internals of the controller to evolve as time passes by but no new percept is available.

Implemented in [TrivialSaccadeController](#), [FixedSaccadeController](#), [FrictionSaccadeController](#), [ThresholdSaccadeController](#), [ThresholdFrictionSaccadeController](#), [SimpleEyeHeadSaccadeController](#), [MonkeyEyeHeadController](#), [Monkey2SaccadeController](#), [MonkeySaccadeController](#), [EyeTrackerSaccadeController](#), [VisBufSaccadeController](#), [ModelEyeHeadController](#), and [ModelWrapperSaccadeController](#).

**10.238.3.13 Point2D SaccadeController::getDecision (const double t)** [virtual]

Get the fixation decided by the controller.

**Parameters:**

*t* should be the current time (in seconds) and should be greater than or equal to the time of the last percept passed to [setPercept\(\)](#). The coordinates of the decided fixation are returned (or (-1, -1) if no decision has been made). If a decision is made but is the same as the previous one, the behavior here is to return (-1, -1), and the caller has the responsibility of holding fixation until a new, valid decision can be reached. Do not overload this method, overload [computeWhenNewDecision\(\)](#) instead if you need to do some computation before returning a fixation.

Definition at line 111 of file SaccadeController.C.

References [computeWhenNewDecision\(\)](#), [Point2D::i](#), and [Point2D::j](#).

**10.238.3.14 Point2D SaccadeController::getDecisionHead (const double t)** [virtual]

Same as [getDecision\(\)](#) but for the head position.

Definition at line 136 of file SaccadeController.C.

References `computeWhenNewDecisionHead()`, `Point2D::i`, and `Point2D::j`.

#### 10.238.3.15 `int SaccadeController::getDqlen () const`

Returns length of decision queue.

Note that the queue may not be full to that length yet

Definition at line 229 of file SaccadeController.C.

#### 10.238.3.16 `int SaccadeController::getDqlenHead () const`

Returns length of head decision queue.

Note that the queue may not be full to that length yet

Definition at line 245 of file SaccadeController.C.

#### 10.238.3.17 `int SaccadeController::getPqlen () const`

Returns length of percept queue.

Note that the queue may not be full to that length yet

Definition at line 197 of file SaccadeController.C.

#### 10.238.3.18 `Point2DT SaccadeController::getPreviousDecision (const unsigned int index = 0) const`

Get a previous decision.

This method allows one to access a previous valid decision. It will not attempt to determine whether a new decision can be made now, like `getDecision()` does. It will simply return the result of what was previously decided (or (-1, -1) if no valid previous decision exists).

##### Parameters:

***index*** indicates which previous decision; a value of zero will return the last decision that was made and returned by `getDecision()`, and increasing values will return increasingly older decisions. if *index* is more than the queue length, (-1, -1) will be returned.

Definition at line 209 of file SaccadeController.C.

Referenced by `ModelEyeHeadController::computeWhenNewDecision()`, `MonkeyEyeHeadController::computeWhenNewDecision()`, `ThresholdFrictionSaccade-`

Controller::computeWhenNewDecision(), and ThresholdSaccadeController::computeWhenNewDecision().

#### 10.238.3.19 **Point2DT** SaccadeController::getPreviousDecisionHead (const unsigned int *index* = 0) const

Same as getPreviousDecision but for the head.

Definition at line 217 of file SaccadeController.C.

#### 10.238.3.20 **WTAwinner** SaccadeController::getPreviousPercept (const unsigned int *index* = 0) const

Get a previous percept.

This method allows one to access a previous percept passed through [setPercept\(\)](#), or (-1, -1) if no valid previous percept exists.

##### Parameters:

*index* indicates which previous percept; a value of zero will return the last percept that was passed to [setPercept\(\)](#), and increasing values will return increasingly older percepts. If *index* is more than the queue length, (-1, -1) will be returned.

Definition at line 185 of file SaccadeController.C.

Referenced by ThresholdSaccadeController::checkPercepts(), MonkeySaccadeController::computeWhenNewDecision(), TrivialSaccadeController::computeWhenNewDecision(), ModelEyeHeadController::computeWhenNewPercept(), ModelWrapperSaccadeController::computeWhenNewPercept(), MonkeySaccadeController::computeWhenNewPercept(), Monkey2SaccadeController::computeWhenNewPercept(), MonkeyEyeHeadController::computeWhenNewPercept(), SimpleEyeHeadSaccadeController::computeWhenNewPercept(), ThresholdFrictionSaccadeController::computeWhenNewPercept(), FrictionSaccadeController::computeWhenNewPercept(), ModelEyeHeadController::computeWhenResetPos(), MonkeySaccadeController::computeWhenResetPos(), MonkeyEyeHeadController::computeWhenResetPos(), ThresholdFrictionSaccadeController::computeWhenResetPos(), ThresholdSaccadeController::computeWhenResetPos(), FrictionSaccadeController::computeWhenResetPos(), FixedSaccadeController::computeWhenResetPos(), and FrictionSaccadeController::evolve().

#### 10.238.3.21 **bool** SaccadeController::haveDecisions () const

Do we already have any decisions queued up?

Definition at line 225 of file SaccadeController.C.

Referenced by `ThresholdFrictionSaccadeController::computeWhenNewDecision()`, and `ThresholdSaccadeController::computeWhenNewDecision()`.

#### **10.238.3.22 `bool SaccadeController::haveDecisionsHead () const`**

Do we already have any head decisions queued up?

Definition at line 241 of file `SaccadeController.C`.

#### **10.238.3.23 `bool SaccadeController::havePercepts () const`**

Do we already have any percepts queued up?

Definition at line 193 of file `SaccadeController.C`.

Referenced by `MonkeySaccadeController::computeWhenNewDecision()`, `ThresholdSaccadeController::computeWhenNewDecision()`, `FrictionSaccadeController::computeWhenNewDecision()`, and `FrictionSaccadeController::evolve()`.

#### **10.238.3.24 `bool SaccadeController::isInBlink () const`**

Are we executing an eye blink?

Definition at line 273 of file `SaccadeController.C`.

Referenced by `FixedSaccadeController::blink()`, `ModelWrapperSaccadeController::computeWhenNewPercept()`, `Monkey2SaccadeController::computeWhenNewPercept()`, `ModelWrapperSaccadeController::evolve()`, `EyeTrackerSaccadeController::evolve()`, `MonkeySaccadeController::evolve()`, and `Monkey2SaccadeController::evolve()`.

#### **10.238.3.25 `bool SaccadeController::isInSaccade () const`**

Are we executing a saccade?

Definition at line 257 of file `SaccadeController.C`.

Referenced by `ModelEyeHeadController::computeWhenResetPos()`, `MonkeyEyeHeadController::computeWhenResetPos()`, `ModelEyeHeadController::evolve()`, `EyeTrackerSaccadeController::evolve()`, `Monkey2SaccadeController::evolve()`, `MonkeyEyeHeadController::evolve()`, `ThresholdFrictionSaccadeController::evolve()`, `ThresholdSaccadeController::evolve()`, `FrictionSaccadeController::evolve()`, and `FixedSaccadeController::saccade()`.

**10.238.3.26 bool SaccadeController::isInSaccadeHead () const**

Are we executing a head "saccade"?

Definition at line 261 of file SaccadeController.C.

**10.238.3.27 void SaccadeController::killDecisions ()**

Kill all previous decisions (empty the decision queue).

Definition at line 237 of file SaccadeController.C.

Referenced by EyeTrackerSaccadeController::evolve().

**10.238.3.28 void SaccadeController::killDecisionsHead ()**

Kill all previous decisions (empty the decision queue).

Definition at line 253 of file SaccadeController.C.

**10.238.3.29 void SaccadeController::killPercepts ()**

Kill all previous percepts (empty the percept queue).

Definition at line 205 of file SaccadeController.C.

**10.238.3.30 void SaccadeController::reset (const bool *recurse* = true)  
[virtual]**

Reset SaccadeController.

See the base function in [ModelComponent.H](#) for info.

Reimplemented from [ModelComponent](#).

Reimplemented in [ThresholdSaccadeController](#).

Definition at line 69 of file SaccadeController.C.

References [ModelComponent::reset\(\)](#).

Referenced by [ThresholdSaccadeController::reset\(\)](#).

**10.238.3.31 void SaccadeController::resetDqlen (const int *len*)**

Reset length of decision queue (contents will be lost).

Definition at line 233 of file SaccadeController.C.

**10.238.3.32 void SaccadeController::resetDqlenHead (const int *len*)**

Reset length of head decision queue (contents will be lost).

Definition at line 249 of file SaccadeController.C.

**10.238.3.33 void SaccadeController::resetPos (const [Point2DT](#) & *p*)  
[virtual]**

Reset to an externally-given position.

This will force the current position to the position given as argument, disregarding any internal physical model. So this may be used to externally drive the SaccadeController to given locations. A percept will be queued up, which will be built from the passed position/time plus default values for the [WTAwinner](#) fields. Do not overload this method, overload computeWhenReset() instead.

Definition at line 97 of file SaccadeController.C.

References computeWhenResetPos(), [Point2DT::p](#), [WTAwinner::p](#), [Point2DT::t](#), and [WTAwinner::t](#).

**10.238.3.34 void SaccadeController::resetPqlen (const int *len*)**

Reset length of percept queue (contents will be lost).

Definition at line 201 of file SaccadeController.C.

Referenced by ThresholdSaccadeController::start1().

**10.238.3.35 void SaccadeController::setBrain ([Brain](#) \* *brain*) [virtual]**

Set a hook to our [Brain](#).

Definition at line 61 of file SaccadeController.C.

**10.238.3.36 void SaccadeController::setPercept (const [WTAwinner](#) & *fix*)  
[virtual]**

Receive a new percept (e.g., winner from saliency map).

This method should be invoked by external callers each time a new percept is available. Do not overload this method, overload [computeWhenNewPercept\(\)](#) instead if you need to perform some computation for each new percept.

Definition at line 84 of file SaccadeController.C.

References computeWhenNewPercept().

Referenced by FrictionSaccadeController::start1().

## 10.238.4 Member Data Documentation

### 10.238.4.1 [Brain\\*](#) [SaccadeController::itsBrain](#) [protected]

hook to our [Brain](#) or NULL

Definition at line 259 of file SaccadeController.H.

The documentation for this class was generated from the following files:

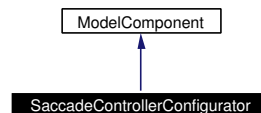
- [SaccadeController.H](#)
- [SaccadeController.C](#)

## 10.239 SaccadeControllerConfigurator Class Reference

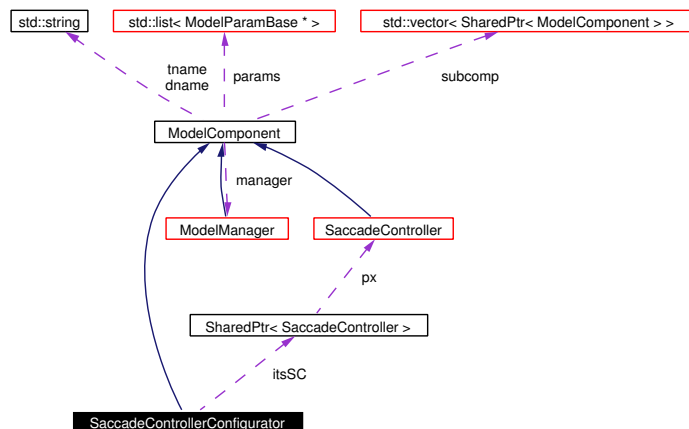
[SaccadeController](#) configurator.

```
#include <SaccadeControllers.H>
```

Inheritance diagram for SaccadeControllerConfigurator:



Collaboration diagram for SaccadeControllerConfigurator:



### Public Member Functions

- [SaccadeControllerConfigurator](#) ([ModelManager](#) &mgr, const [std::string](#) &descrName="Saccade Controller Configurator", const [std::string](#) &tagName="SaccadeControllerConfigurator")

*Constructor.*

- virtual [~SaccadeControllerConfigurator](#) ()

*destructor*

- virtual void [exportOptions](#) (const int whatToExport, const bool recurse=true)



Export some of our [ModelParam](#) parameters as command-line-options.

- [SharedPtr](#)< [SaccadeController](#) > [getSC](#) () const  
Get the chosen [SaccadeController](#).
- virtual bool [setModelParamString](#) (const [std::string](#) &name, const [std::string](#) &value, const bool recurse=false, const bool warn=true)  
Intercept people changing our [itsSacCtrlName](#) [ModelParam](#).

## Protected Attributes

- [ModelParam](#)< [std::string](#) > [itsSacCtrlType](#)  
Type of the controller.

### 10.239.1 Detailed Description

[SaccadeController](#) configurator.

This will export the `--sc-type=XX` command-line option and will instantiate a [SaccadeController](#) of the desired type as the option gets assigned a value. As this happens, new options may become available in the command-line. To see them, use `--help` AFTER you have chosen the name of the controller to use. The current [SaccadeController](#) may be retrieved using [getSC\(\)](#).

Definition at line 54 of file `SaccadeControllers.H`.

### 10.239.2 Constructor & Destructor Documentation

#### 10.239.2.1 [SaccadeControllerConfigurator::SaccadeControllerConfigurator](#) ([ModelManager](#) & mgr, const [std::string](#) & descrName = "Saccade Controller Configurator", const [std::string](#) & tagName = "SaccadeControllerConfigurator")

Constructor.

Definition at line 51 of file `SaccadeControllers.C`.

References `OPTEXP_NONE`.

#### 10.239.2.2 [SaccadeControllerConfigurator::~~SaccadeControllerConfigurator](#) () [virtual]

destructor

Definition at line 60 of file SaccadeControllers.C.

### 10.239.3 Member Function Documentation

#### 10.239.3.1 `void SaccadeControllerConfigurator::exportOptions (const int whatToExport, const bool recurse = true) [virtual]`

Export some of our [ModelParam](#) parameters as command-line-options.

See the base function in [ModelComponent.H](#) for info. Actually, here we just keep a copy of *whatToExport* and we will call [exportOptions\(\)](#) on our [SaccadeController](#) once we know its type and instantiate it.

Reimplemented from [ModelComponent](#).

Definition at line 64 of file SaccadeControllers.C.

References [ModelComponent::exportOptions\(\)](#), [itsSacCtrlType](#), [OPTEXP\\_CORE](#), and [ModelManager::requestOption\(\)](#).

#### 10.239.3.2 `SharedPtr< SaccadeController > SaccadeControllerConfigurator::getSC () const`

Get the chosen [SaccadeController](#).

You should call this during [start\(\)](#) of the [ModelComponent](#) that needs the [SaccadeController](#). CAUTION: this may be a [SharedPtr\(NULL\)](#) if no controller has been chosen.

Definition at line 80 of file SaccadeControllers.C.

#### 10.239.3.3 `bool SaccadeControllerConfigurator::setModelParamString (const std::string & name, const std::string & value, const bool recurse = false, const bool warn = true) [virtual]`

Intercept people changing our [itsSacCtrlName](#) [ModelParam](#).

See [ModelComponent.H](#); as parsing the command-line or reading a config file sets our name, we'll also here instantiate a controller of the proper type (and export its options)

Reimplemented from [ModelComponent](#).

Definition at line 85 of file SaccadeControllers.C.

References [ModelComponent::addSubComponent\(\)](#), [SharedPtr< SaccadeController >::get\(\)](#), [itsSacCtrlType](#), [ModelComponent::removeSubComponent\(\)](#), [SharedPtr< SaccadeController >::reset\(\)](#), and [ModelComponent::setModelParamString\(\)](#).

## 10.239.4 Member Data Documentation

### 10.239.4.1 [ModelParam<std::string>](#) [SaccadeControllerConfigurator::itsSaccCtrlType](#) [protected]

Type of the controller.

Definition at line 91 of file SaccadeControllers.H.

Referenced by [exportOptions\(\)](#), and [setModelParamString\(\)](#).

The documentation for this class was generated from the following files:

- [SaccadeControllers.H](#)
- [SaccadeControllers.C](#)

## 10.240 SafeCopyHelper< T, isTrivial > Struct Template Reference

Helper template struct for pointer version of [safecopy\(\)](#).

```
#include <safecopy.H>
```

### Static Public Member Functions

- void **copy** (T \*dest, const T \*src, unsigned int n)

#### 10.240.1 Detailed Description

**template<typename T, bool isTrivial = TypeTraits<T>::isTrivial> struct SafeCopyHelper< T, isTrivial >**

Helper template struct for pointer version of [safecopy\(\)](#).

Definition at line 66 of file safecopy.H.

The documentation for this struct was generated from the following file:

- [safecopy.H](#)

## 10.241 SafeCopyHelper< T, true > Struct Template Reference

Specialized implementation for trivial types.

```
#include <safecopy.H>
```

### Static Public Member Functions

- void **copy** (T \*dest, const T \*src, unsigned int n)

#### 10.241.1 Detailed Description

```
template<typename T> struct SafeCopyHelper< T, true >
```

Specialized implementation for trivial types.

Definition at line 74 of file safecopy.H.

The documentation for this struct was generated from the following file:

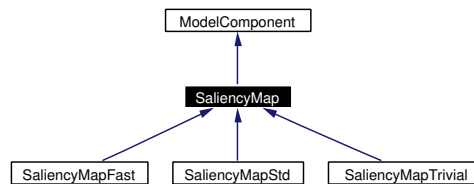
- [safecopy.H](#)

## 10.242 SaliencyMap Class Reference

The saliency map base class.

```
#include <SaliencyMap.H>
```

Inheritance diagram for SaliencyMap:



Collaboration diagram for SaliencyMap:

### Public Member Functions

#### Constructor, destructor, and reset

- `SaliencyMap (ModelManager &mgr, const std::string &descrName="Saliency Map", const std::string &tagName="SaliencyMap")`  
*Initialized constructor.*
- `virtual void exportOptions (const int whatToExport, const bool recurse=true)`  
*Export some of our ModelParam parameters as command-line-options.*
- `virtual ~SaliencyMap ()`  
*Destructor.*
- `virtual void reset ()=0`  
*Reset to initial state just after construction.*

#### Input, evolve and output

- `virtual void input (const Image< float > &in)=0`  
*Set new input currents for all neurons.*
- `virtual int getWidth () const=0`  
*Get width of saliency map.*

- virtual int [getHeight](#) () const=0  
*Get height of saliency map.*
- virtual [Image](#)< float > [getV](#) (const bool normalize=false) const=0  
*Return all our membrane potential voltages as an [Image](#)<float>.*
- virtual float [getV](#) (const [Point2D](#) &p) const=0  
*Access function using [Point2D](#).*
- virtual void [evolve](#) (const double t)=0  
*Integrate inputs until time t (in s) and update membrane potentials.*

### Inhibition of return

- virtual void [depress](#) (const [Point2D](#) &winner, const double &pampl, const double &mampl, const double &psdev, const double &msdev)=0  
*difference-of-Gaussians inhibition of return*
- virtual void [depress](#) (const [Point2D](#) &winner, const double &ampl, const [Image](#)< byte > &objectMask)=0  
*object-based inhibition of return*

### Saccadic and blink suppression

- virtual void [saccadicSuppression](#) (const bool on)=0  
*Turn saccadic suppression on/off.*
- virtual void [blinkSuppression](#) (const bool on)=0  
*Turn blink suppression on/off.*

### Results

- virtual void [saveResults](#) (const [SharedPtr](#)< [OutputFrameSeries](#) > &ofs)  
*Save our various results.*

### Protected Member Functions

- double [w](#) (const double &d, const double &ampl, const double &sdev)  
*Helper function to compute inhibition of return.*
- double [w](#) (const [Point2D](#) &p1, const [Point2D](#) &p2, const double &pampl, const double &mampl, const double &psdev, const double &msdev)  
*Helper function to compute inhibition of return.*

## Protected Attributes

- [ModelParam](#)< bool > [itsSaveResults](#)  
*Save our internals when [saveResults\(\)](#) is called?*
- [ModelParam](#)< bool > [itsSaveCumResults](#)  
*Save our cumulative internals when [saveResults\(\)](#) is called?*
- [ModelParam](#)< float > [itsSalmapFactor](#)  
*Multiply SM by a factor rather than normalize when doing [getV\(\)](#)?*
- [ModelParam](#)< bool > [itsUseSaccadicSuppression](#)  
*use saccadic suppression?*
- [ModelParam](#)< bool > [itsUseBlinkSuppression](#)  
*use blink suppression?*
- [Image](#)< float > [itsCumMap](#)  
*Cumulative map so far. Cumulates each time [saveResults\(\)](#) is called.*

### 10.242.1 Detailed Description

The saliency map base class.

This is a 2D salience map. It is just a base class with virtual function definitions. Various saliency maps are available below, which provide implementations for these virtual functions. The saliency map receives input through the [input\(\)](#) function, and its current state (membrane potential of neurons) can be retrieved using [getV\(\)](#). Various implementations will typically derive from class [Image](#) and inherit all the standard methods of class [Image](#). The user has the responsibility of calling [evolve\(\)](#) often to ensure that possible difference equations in the map's neurons run well.

Definition at line 58 of file SaliencyMap.H.

### 10.242.2 Constructor & Destructor Documentation

#### 10.242.2.1 SaliencyMap::SaliencyMap ([ModelManager](#) & mgr, const [std::string](#) & descrName = "Saliency Map", const [std::string](#) & tagName = "SaliencyMap")

Initialized constructor.

The SM will be resized and initialized the first time [input\(\)](#) is called

Definition at line 52 of file SaliencyMap.C.



### 10.242.2.2 SaliencyMap::~SaliencyMap () [virtual]

Destructor.

Definition at line 63 of file SaliencyMap.C.

## 10.242.3 Member Function Documentation

### 10.242.3.1 virtual void SaliencyMap::blinkSuppression (const bool *on*) [pure virtual]

Turn blink suppression on/off.

Implemented in [SaliencyMapStd](#), [SaliencyMapTrivial](#), and [SaliencyMapFast](#).

### 10.242.3.2 virtual void SaliencyMap::depress (const [Point2D](#) & *winner*, const double & *ampl*, const [Image](#)< [byte](#) > & *objectMask*) [pure virtual]

object-based inhibition of return

The object mask defines where in the image the inhibition will be applied. The winner coordinates and ampl are used to calibrate the inhibition strength

Implemented in [SaliencyMapStd](#), [SaliencyMapTrivial](#), and [SaliencyMapFast](#).

### 10.242.3.3 virtual void SaliencyMap::depress (const [Point2D](#) & *winner*, const double & *pampl*, const double & *mampl*, const double & *psdev*, const double & *msdev*) [pure virtual]

difference-of-Gaussians inhibition of return

The positive Gaussian is defined by its amplitude pampl and width psdev; the negative Gaussian by mampl and msdev. Both centered at winner.

Implemented in [SaliencyMapStd](#), [SaliencyMapTrivial](#), and [SaliencyMapFast](#).

### 10.242.3.4 virtual void SaliencyMap::evolve (const double *t*) [pure virtual]

Integrate inputs until time t (in s) and update membrane potentials.

Implemented in [SaliencyMapStd](#), [SaliencyMapTrivial](#), and [SaliencyMapFast](#).

### 10.242.3.5 **void SaliencyMap::exportOptions (const int *whatToExport*, const bool *recurse* = true) [virtual]**

Export some of our [ModelParam](#) parameters as command-line-options.

See the base function in [ModelComponent.H](#) for info. A base implementation is provided that will only export the base class options (see the [ModelParam](#) definitions in the protected section below).

Reimplemented from [ModelComponent](#).

Reimplemented in [SaliencyMapStd](#), and [SaliencyMapFast](#).

Definition at line 67 of file [SaliencyMap.C](#).

References [ModelComponent::exportOptions\(\)](#), [itsSalmapFactor](#), [itsSaveCumResults](#), [itsUseBlinkSuppression](#), [itsUseSaccadicSuppression](#), [OPTEXP\\_CORE](#), [OPTEXP\\_SAVE](#), and [ModelManager::requestOption\(\)](#).

Referenced by [SaliencyMapStd::exportOptions\(\)](#).

### 10.242.3.6 **virtual int SaliencyMap::getHeight () const [pure virtual]**

Get height of saliency map.

Implemented in [SaliencyMapStd](#), [SaliencyMapTrivial](#), and [SaliencyMapFast](#).

### 10.242.3.7 **virtual float SaliencyMap::getV (const [Point2D](#) & p) const [pure virtual]**

Access function using [Point2D](#).

Implemented in [SaliencyMapStd](#), [SaliencyMapTrivial](#), and [SaliencyMapFast](#).

### 10.242.3.8 **virtual [Image](#)<float> SaliencyMap::getV (const bool *normalize* = false) const [pure virtual]**

Return all our membrane potential voltages as an [Image](#)<float>.

The current values are possibly normalized to 0..255. The normalization will use itsSalmapFactor if it is non-null

Implemented in [SaliencyMapStd](#), [SaliencyMapTrivial](#), and [SaliencyMapFast](#).

Referenced by [saveResults\(\)](#).

### 10.242.3.9 **virtual int SaliencyMap::getWidth () const [pure virtual]**

Get width of saliency map.

Implemented in [SaliencyMapStd](#), [SaliencyMapTrivial](#), and [SaliencyMapFast](#).

#### 10.242.3.10 **virtual void SaliencyMap::input (const [Image](#)< float > & in)** [pure virtual]

Set new input currents for all neurons.

This should initialize and resize the saliency map if the saliency map is currently uninitialized (e.g., just after construction or [reset\(\)](#)).

Implemented in [SaliencyMapStd](#), [SaliencyMapTrivial](#), and [SaliencyMapFast](#).

#### 10.242.3.11 **virtual void SaliencyMap::reset ()** [pure virtual]

Reset to initial state just after construction.

Implemented in [SaliencyMapStd](#), [SaliencyMapTrivial](#), and [SaliencyMapFast](#).

#### 10.242.3.12 **virtual void SaliencyMap::saccadicSuppression (const bool on)** [pure virtual]

Turn saccadic suppression on/off.

Implemented in [SaliencyMapStd](#), [SaliencyMapTrivial](#), and [SaliencyMapFast](#).

#### 10.242.3.13 **void SaliencyMap::saveResults (const [SharedPtr](#)< [OutputFrameSeries](#) > & ofs)** [virtual]

Save our various results.

A default implementation is provided, which just calls [getV\(\)](#) and saves that map with a file name prefix of "SM".

#### Parameters:

*ofs* will figure out file name/format and save our results.

Definition at line 105 of file [SaliencyMap.C](#).

References [getV\(\)](#), [ModelParam](#)< bool >::getVal(), [Image](#)< float >::initialized(), [itsCumMap](#), and [itsSaveCumResults](#).

#### 10.242.3.14 **double SaliencyMap::w (const [Point2D](#) & p1, const [Point2D](#) & p2, const double & paml, const double & maml, const double & psdev, const double & msdev)** [protected]

Helper function to compute inhibition of return.

Definition at line 94 of file SaliencyMap.C.

References Point2D::i, Point2D::j, SQR(), sqrt(), and w().

#### 10.242.3.15 **double SaliencyMap::w (const double & d, const double & ampl, const double & sdev) [protected]**

Helper function to compute inhibition of return.

Definition at line 86 of file SaliencyMap.C.

References exp().

Referenced by SaliencyMapFast::depress(), SaliencyMapTrivial::depress(), SaliencyMapStd::depress(), and w().

### 10.242.4 Member Data Documentation

#### 10.242.4.1 **Image<float> SaliencyMap::itsCumMap [protected]**

Cumulative map so far. Cumulates each time [saveResults\(\)](#) is called.

This is computed only if [itsSaveCumResults\(\)](#) is true

Definition at line 187 of file SaliencyMap.H.

Referenced by [saveResults\(\)](#).

#### 10.242.4.2 **ModelParam<float> SaliencyMap::itsSalmapFactor [protected]**

Multiply SM by a factor rather than normalize when doing [getV\(\)](#)?

Definition at line 170 of file SaliencyMap.H.

Referenced by [exportOptions\(\)](#).

#### 10.242.4.3 **ModelParam<bool> SaliencyMap::itsSaveCumResults [protected]**

Save our cumulative internals when [saveResults\(\)](#) is called?

Definition at line 167 of file SaliencyMap.H.

Referenced by [exportOptions\(\)](#), and [saveResults\(\)](#).

**10.242.4.4   [ModelParam<bool>](#) [SaliencyMap::itsSaveResults](#)   [protected]**

Save our internals when [saveResults\(\)](#) is called?

Definition at line 164 of file SaliencyMap.H.

**10.242.4.5   [ModelParam<bool>](#) [SaliencyMap::itsUseBlinkSuppression](#)   [protected]**

use blink suppression?

Definition at line 176 of file SaliencyMap.H.

Referenced by [exportOptions\(\)](#).

**10.242.4.6   [ModelParam<bool>](#) [SaliencyMap::itsUseSaccadicSuppression](#)   [protected]**

use saccadic suppression?

Definition at line 173 of file SaliencyMap.H.

Referenced by [exportOptions\(\)](#).

The documentation for this class was generated from the following files:

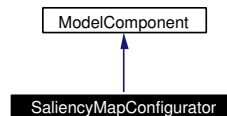
- [SaliencyMap.H](#)
- [SaliencyMap.C](#)

## 10.243 SaliencyMapConfigurator Class Reference

[SaliencyMap](#) configurator.

```
#include <SaliencyMap.H>
```

Inheritance diagram for SaliencyMapConfigurator:



Collaboration diagram for SaliencyMapConfigurator:

### Public Member Functions

- [SaliencyMapConfigurator](#) ([ModelManager](#) &mgr, const [std::string](#) &descr-Name="Saliency Map Configurator", const [std::string](#) &tagName="Saliency-MapConfigurator")

*Constructor.*

- virtual [~SaliencyMapConfigurator](#) ()

*destructor*

- virtual void [exportOptions](#) (const int whatToExport, const bool recurse=true)

*Export some of our [ModelParam](#) parameters as command-line-options.*

- [SharedPtr](#)< [SaliencyMap](#) > [getSM](#) () const

*Get the chosen SM.*

### Protected Member Functions

- virtual bool [setModelParamString](#) (const [std::string](#) &name, const [std::string](#) &value, const bool recurse=false, const bool warn=true)

*Intercept people changing our [ModelParam](#).*

### Protected Attributes

- [ModelParam](#)< [std::string](#) > [itsSMtype](#)

*type of saliency map*

### 10.243.1 Detailed Description

[SaliencyMap](#) configurator.

This will export the `-sm-type=XX` command-line option and will instantiate an SM of the desired type as the option gets assigned a value. As this happens, new options may become available in the command-line. To see them, use `-help` AFTER you have chosen the type to use. The current SM may be retrieved using [getSM\(\)](#).

Definition at line 203 of file `SaliencyMap.H`.

### 10.243.2 Constructor & Destructor Documentation

**10.243.2.1** `SaliencyMapConfigurator::SaliencyMapConfigurator`  
([ModelManager](#) & *mgr*, const `std::string` & *descrName* =  
"Saliency Map Configurator", const `std::string` & *tagName* =  
"SaliencyMapConfigurator")

Constructor.

Definition at line 125 of file `SaliencyMap.C`.

References `OPTEXP_NONE`.

**10.243.2.2** `SaliencyMapConfigurator::~SaliencyMapConfigurator` ()  
[virtual]

destructor

Definition at line 134 of file `SaliencyMap.C`.

### 10.243.3 Member Function Documentation

**10.243.3.1** `void SaliencyMapConfigurator::exportOptions` (const int  
*whatToExport*, const bool *recurse* = true) [virtual]

Export some of our [ModelParam](#) parameters as command-line-options.

See the base function in [ModelComponent.H](#) for info. Actually, here we just keep a copy of *whatToExport* and we will call [exportOptions\(\)](#) on our SM once we know its type and instantiate it.

Reimplemented from [ModelComponent](#).

Definition at line 138 of file SaliencyMap.C.

References `ModelComponent::exportOptions()`, `itsSMtype`, `OPTEXP_CORE`, and `ModelManager::requestOption()`.

#### 10.243.3.2 `SharedPtr< SaliencyMap > SaliencyMapConfigurator::getSM ()` `const`

Get the chosen SM.

You should call this during `start()` of the `ModelComponent` that needs the SM. CAUTION: this may be a `SharedPtr(NULL)` if no SM type has been chosen.

Definition at line 154 of file SaliencyMap.C.

#### 10.243.3.3 `bool SaliencyMapConfigurator::setModelParamString (const` `std::string & name, const std::string & value, const bool recurse =` `false, const bool warn = true) [protected, virtual]`

Intercept people changing our `ModelParam`.

See `ModelComponent.H`; as parsing the command-line or reading a config file sets our name, we'll also here instantiate a controller of the proper type (and export its options)

Reimplemented from `ModelComponent`.

Definition at line 158 of file SaliencyMap.C.

References `ModelComponent::addSubComponent()`, `SharedPtr< SaliencyMap >::get()`, `itsSMtype`, `ModelComponent::removeSubComponent()`, `SharedPtr< SaliencyMap >::reset()`, and `ModelComponent::setModelParamString()`.

### 10.243.4 Member Data Documentation

#### 10.243.4.1 `ModelParam<std::string> SaliencyMapConfigurator::itsSMtype` `[protected]`

type of saliency map

Definition at line 231 of file SaliencyMap.H.

Referenced by `exportOptions()`, and `setModelParamString()`.

The documentation for this class was generated from the following files:

- `SaliencyMap.H`
- `SaliencyMap.C`

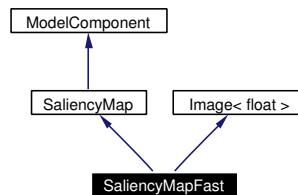


## 10.244 SaliencyMapFast Class Reference

The fast saliency map: a 2D array of floats.

```
#include <SaliencyMap.H>
```

Inheritance diagram for SaliencyMapFast:



Collaboration diagram for SaliencyMapFast:

### Public Member Functions

- [SaliencyMapFast](#) ([ModelManager](#) &mgr, const [std::string](#) &descr-Name="Saliency Map Fast", const [std::string](#) &tagName="SaliencyMapFast")  
*Initialized constructor.*
- virtual void [exportOptions](#) (const int whatToExport, const bool recurse=true)  
*Export some of our [ModelParam](#) parameters as command-line-options.*
- virtual [~SaliencyMapFast](#) ()  
*Destructor.*
- virtual void [reset](#) ()  
*Reset to initial state just after construction.*
- virtual void [input](#) (const [Image<float>](#) &in)  
*Set new input currents for all neurons.*
- virtual [Image<float>](#) [getV](#) (const bool normalize=false) const  
*Return all our membrane potential voltages as an [Image<float>](#).*
- virtual float [getV](#) (const [Point2D](#) &p) const  
*Access function using [Point2D](#).*
- virtual int [getWidth](#) () const

*Get width of saliency map.*

- virtual int [getHeight](#) () const  
*Get height of saliency map.*
- virtual void [evolve](#) (const double t)  
*Integrate inputs until time t (in s) and update membrane potential.*
- virtual void [depress](#) (const [Point2D](#) &winner, const double &pampl, const double &mampl, const double &psdev, const double &msdev)  
*difference-of-Gaussians inhibition of return*
- virtual void [depress](#) (const [Point2D](#) &winner, const double &ampl, const [Image](#)< byte > &objectMask)  
*object-based inhibition of return*
- virtual void [saccadicSuppression](#) (const bool on)  
*Turn saccadic suppression on/off.*
- virtual void [blinkSuppression](#) (const bool on)  
*Turn blink suppression on/off.*

## Protected Attributes

- [ModelParam](#)< double > [itsTimeStep](#)  
*Simulation time step, in seconds.*
- [ModelParam](#)< float > [itsInputCoeff](#)  
*Coeff by which new inputs replace the current SM at each time step (0..1).*

### 10.244.1 Detailed Description

The fast saliency map: a 2D array of floats.

This is a 2D salience map. It is an [Image](#)<float> and inherits all the standard methods of class [Image](#). To avoid confusion, we here add explicit [input\(\)](#) and [getV\(\)](#) methods. The user has the responsibility of calling [evolve\(\)](#) often to ensure that the saliency map evolves smoothly over time.

Definition at line 420 of file SaliencyMap.H.

## 10.244.2 Constructor & Destructor Documentation

### 10.244.2.1 SaliencyMapFast::SaliencyMapFast ([ModelManager](#) & *mgr*, const [std::string](#) & *descrName* = "Saliency Map Fast", const [std::string](#) & *tagName* = "SaliencyMapFast")

Initialized constructor.

The SM will be resized and initialized the first time [input\(\)](#) is called

Definition at line 501 of file SaliencyMap.C.

### 10.244.2.2 SaliencyMapFast::~SaliencyMapFast () [[virtual](#)]

Destructor.

Definition at line 524 of file SaliencyMap.C.

## 10.244.3 Member Function Documentation

### 10.244.3.1 void SaliencyMapFast::blinkSuppression (const bool *on*) [[virtual](#)]

Turn blink suppression on/off.

Implements [SaliencyMap](#).

Definition at line 589 of file SaliencyMap.C.

References [ModelParam< bool >::getVal\(\)](#).

### 10.244.3.2 void SaliencyMapFast::depress (const [Point2D](#) & *winner*, const double & *ampl*, const [Image](#)< [byte](#) > & *objectMask*) [[virtual](#)]

object-based inhibition of return

The object mask defines where in the image the inhibition will be applied. The winner coordinates and ampl are used to calibrate the inhibition strength

Implements [SaliencyMap](#).

Definition at line 567 of file SaliencyMap.C.

References [Image< float >::rectify\(\)](#).

**10.244.3.3** `void SaliencyMapFast::depress (const Point2D & winner, const double & paml, const double & mampl, const double & psdev, const double & msdev) [virtual]`

difference-of-Gaussians inhibition of return

The positive Gaussian is defined by its amplitude paml and width psdev; the negative Gaussian by mampl and msdev. Both centered at winner.

Implements [SaliencyMap](#).

Definition at line 546 of file SaliencyMap.C.

References `Image< float >::beginw()`, `getHeight()`, `getWidth()`, `Point2D::i`, `Point2D::j`, and `SaliencyMap::w()`.

**10.244.3.4** `void SaliencyMapFast::evolve (const double t) [virtual]`

Integrate inputs until time t (in s) and update membrane potential.

Implements [SaliencyMap](#).

Definition at line 603 of file SaliencyMap.C.

References `Image< float >::addWeighted()`, `ASSERT`, `ModelParam< float >::getVal()`, `ModelParam< double >::getVal()`, `Image< float >::initialized()`, and `itsInputCoeff`.

**10.244.3.5** `void SaliencyMapFast::exportOptions (const int whatToExport, const bool recurse = true) [virtual]`

Export some of our [ModelParam](#) parameters as command-line-options.

See the base function in [ModelComponent.H](#) for info.

Reimplemented from [SaliencyMap](#).

Definition at line 511 of file SaliencyMap.C.

References `ModelComponent::exportOptions()`, `itsInputCoeff`, `OPTEXP_CORE`, and `ModelManager::requestOption()`.

**10.244.3.6** `int SaliencyMapFast::getHeight () const [virtual]`

Get height of saliency map.

Implements [SaliencyMap](#).

Definition at line 625 of file SaliencyMap.C.

References `Image< T >::getHeight()`.

Referenced by `depress()`.

#### 10.244.3.7 `float SaliencyMapFast::getV (const Point2D & p) const` [virtual]

Access function using [Point2D](#).

Implements [SaliencyMap](#).

Definition at line 613 of file `SaliencyMap.C`.

References `Image< float >::getVal()`.

#### 10.244.3.8 `Image< float > SaliencyMapFast::getV (const bool normalize = false) const` [virtual]

Return all our membrane potential voltages as an [Image<float>](#).

Implements [SaliencyMap](#).

Definition at line 535 of file `SaliencyMap.C`.

References `ModelParam< float >::getVal()`, and `Image< T >::normalize()`.

#### 10.244.3.9 `int SaliencyMapFast::getWidth () const` [virtual]

Get width of saliency map.

Implements [SaliencyMap](#).

Definition at line 621 of file `SaliencyMap.C`.

References `Image< T >::getWidth()`.

Referenced by `depress()`.

#### 10.244.3.10 `void SaliencyMapFast::input (const Image< float > & in)` [virtual]

Set new input currents for all neurons.

Implements [SaliencyMap](#).

Definition at line 528 of file `SaliencyMap.C`.

References `Image< T >::getDims()`, `Image< float >::initialized()`, and `Image< float >::resize()`.

**10.244.3.11 void SaliencyMapFast::reset () [virtual]**

Reset to initial state just after construction.

Implements [SaliencyMap](#).

Definition at line 617 of file SaliencyMap.C.

References `Image< float >::freeMem()`.

**10.244.3.12 void SaliencyMapFast::saccadicSuppression (const bool on) [virtual]**

Turn saccadic suppression on/off.

The behavior here is that turning saccadic suppression on will open the inhibitory conductances of the SM neurons just like inhibition of return does. This function should be called repeatedly during suppression as, otherwise, the inhibitory conductances have a natural tendency to fade away (as is needed for IOR to pregressively fade away). When this function is called to turn saccadic suppression off, the inhibitory conductances are violently reset to zero. So, for full saccadic suppression, call `SM.saccadicSuppression(true)` at every time step during the suppression time interval, then call `SM.saccadicSuppression(false)` once at the end of the suppression period. Calling `SM.saccadicSuppression(true)` less often (or just once at the start of a saccade) will yield milder suppression.

Implements [SaliencyMap](#).

Definition at line 576 of file SaliencyMap.C.

References `ModelParam< bool >::getVal()`.

**10.244.4 Member Data Documentation****10.244.4.1 [ModelParam<float>](#) [SaliencyMapFast::itsInputCoeff](#) [protected]**

Coeff by which new inputs replace the current SM at each time step (0..1).

Definition at line 498 of file SaliencyMap.H.

Referenced by `evolve()`, and `exportOptions()`.

**10.244.4.2 [ModelParam<double>](#) [SaliencyMapFast::itsTimeStep](#) [protected]**

Simulation time step, in seconds.

Definition at line 495 of file SaliencyMap.H.

The documentation for this class was generated from the following files:

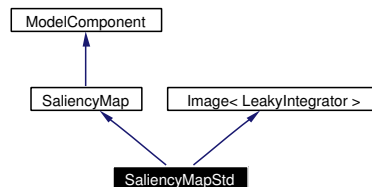
- [SaliencyMap.H](#)
- [SaliencyMap.C](#)

## 10.245 SaliencyMapStd Class Reference

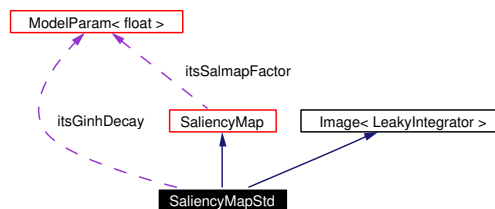
The standard saliency map: a 2D array of [LeakyIntegrator](#) neurons.

```
#include <SaliencyMap.H>
```

Inheritance diagram for SaliencyMapStd:



Collaboration diagram for SaliencyMapStd:



### Public Member Functions

- [SaliencyMapStd](#) ([ModelManager](#) &mgr, const [std::string](#) &descr-Name="Saliency Map Std", const [std::string](#) &tagName="SaliencyMapStd")  
*Initialized constructor.*
- virtual void [exportOptions](#) (const int whatToExport, const bool recurse=true)  
*Export some of our [ModelParam](#) parameters as command-line-options.*
- virtual [~SaliencyMapStd](#) ()  
*Destructor.*
- virtual void [reset](#) ()  
*Reset to initial state just after construction.*
- virtual void [input](#) (const [Image](#)< float > &in)



*Set new input currents for all neurons.*

- virtual int [getWidth](#) () const  
*Get width of saliency map.*
- virtual int [getHeight](#) () const  
*Get height of saliency map.*
- virtual [Image](#)< float > [getV](#) (const bool normalize=false) const  
*Return all our membrane potential voltages as an [Image](#)<float>.*
- virtual float [getV](#) (const [Point2D](#) &p) const  
*Access function using [Point2D](#).*
- virtual void [evolve](#) (const double t)  
*Integrate inputs until time t (in s) and update membrane potential.*
- virtual void [depress](#) (const [Point2D](#) &winner, const double &pampl, const double &mampl, const double &psdev, const double &msdev)  
*difference-of-Gaussians inhibition of return*
- virtual void [depress](#) (const [Point2D](#) &winner, const double &ampl, const [Image](#)< byte > &objectMask)  
*object-based inhibition of return*
- virtual void [saccadicSuppression](#) (const bool on)  
*Turn saccadic suppression on/off.*
- virtual void [blinkSuppression](#) (const bool on)  
*Turn blink suppression on/off.*

## Protected Attributes

- [ModelParam](#)< float > [itsGinhDecay](#)  
*decay for IOR inhibition*

### 10.245.1 Detailed Description

The standard saliency map: a 2D array of [LeakyIntegrator](#) neurons.

This is a 2D salience map. It is an [Image<LeakyIntegrator>](#) and inherits all the standard methods of class [Image](#). To avoid confusion, we here add explicit [input\(\)](#) and [getV\(\)](#) methods (rather than providing conversion functions between [LeakyIntegrator](#) and float, which could make the process more transparent but also possibly more confusing). The user has the responsibility of calling [evolve\(\)](#) often to ensure that the difference equations in the [LeakyIntegrator](#) neurons run well.

Definition at line 259 of file SaliencyMap.H.

## 10.245.2 Constructor & Destructor Documentation

### 10.245.2.1 SaliencyMapStd::SaliencyMapStd ([ModelManager](#) & mgr, const [std::string](#) & descrName = "Saliency Map Std", const [std::string](#) & tagName = "SaliencyMapStd")

Initialized constructor.

The SM will be resized and initialized the first time [input\(\)](#) is called

Definition at line 208 of file SaliencyMap.C.

### 10.245.2.2 SaliencyMapStd::~~SaliencyMapStd () [virtual]

Destructor.

Definition at line 227 of file SaliencyMap.C.

## 10.245.3 Member Function Documentation

### 10.245.3.1 void SaliencyMapStd::blinkSuppression (const bool on) [virtual]

Turn blink suppression on/off.

Implements [SaliencyMap](#).

Definition at line 332 of file SaliencyMap.C.

References [Image< LeakyIntegrator >::beginw\(\)](#), [Image< LeakyIntegrator >::endw\(\)](#), and [ModelParam< bool >::getVal\(\)](#).

### 10.245.3.2 void SaliencyMapStd::depress (const [Point2D](#) & winner, const double & ampl, const [Image< byte >](#) & objectMask) [virtual]

object-based inhibition of return

The object mask defines where in the image the inhibition will be applied. The winner coordinates and ampl are used to calibrate the inhibition strength

Implements [SaliencyMap](#).

Definition at line 287 of file SaliencyMap.C.

References `Image< LeakyIntegrator >::beginw()`, `getHeight()`, `Image< T >::getVal()`, `getWidth()`, `Point2D::i`, and `Point2D::j`.

**10.245.3.3** `void SaliencyMapStd::depress (const Point2D & winner, const double & pampl, const double & mampl, const double & psdev, const double & msdev) [virtual]`

difference-of-Gaussians inhibition of return

The positive Gaussian is defined by its amplitude pampl and width psdev; the negative Gaussian by mampl and msdev. Both centered at winner.

Implements [SaliencyMap](#).

Definition at line 267 of file SaliencyMap.C.

References `Image< LeakyIntegrator >::beginw()`, `getHeight()`, `getWidth()`, `Point2D::i`, `Point2D::j`, and `SaliencyMap::w()`.

**10.245.3.4** `void SaliencyMapStd::evolve (const double t) [virtual]`

Integrate inputs until time t (in s) and update membrane potential.

Implements [SaliencyMap](#).

Definition at line 357 of file SaliencyMap.C.

References `ASSERT`, `Image< LeakyIntegrator >::beginw()`, `Image< LeakyIntegrator >::endw()`, and `Image< LeakyIntegrator >::initialized()`.

**10.245.3.5** `void SaliencyMapStd::exportOptions (const int whatToExport, const bool recurse = true) [virtual]`

Export some of our [ModelParam](#) parameters as command-line-options.

Reimplemented from [SaliencyMap](#).

Definition at line 215 of file SaliencyMap.C.

References `SaliencyMap::exportOptions()`, `OPTEXP_CORE`, and `ModelManager::requestOption()`.

**10.245.3.6 int SaliencyMapStd::getHeight () const [virtual]**

Get height of saliency map.

Implements [SaliencyMap](#).

Definition at line 380 of file SaliencyMap.C.

References [Image< T >::getHeight\(\)](#).

Referenced by [depress\(\)](#).

**10.245.3.7 float SaliencyMapStd::getV (const [Point2D](#) & p) const [virtual]**

Access function using [Point2D](#).

Implements [SaliencyMap](#).

Definition at line 368 of file SaliencyMap.C.

References [LeakyIntegrator::getV\(\)](#), and [Image< LeakyIntegrator >::getVal\(\)](#).

**10.245.3.8 [Image](#)< float > SaliencyMapStd::getV (const bool *normalize* = false) const [virtual]**

Return all our membrane potential voltages as an [Image<float>](#).

Implements [SaliencyMap](#).

Definition at line 248 of file SaliencyMap.C.

References [Image< LeakyIntegrator >::begin\(\)](#), [Image< T >::beginw\(\)](#), [Image< LeakyIntegrator >::end\(\)](#), [Image< LeakyIntegrator >::getDims\(\)](#), [ModelParam< float >::getVal\(\)](#), [NO\\_INIT](#), and [Image< T >::normalize\(\)](#).

**10.245.3.9 int SaliencyMapStd::getWidth () const [virtual]**

Get width of saliency map.

Implements [SaliencyMap](#).

Definition at line 376 of file SaliencyMap.C.

References [Image< T >::getWidth\(\)](#).

Referenced by [depress\(\)](#).

### 10.245.3.10 void SaliencyMapStd::input (const [Image](#)< float > & in) [virtual]

Set new input currents for all neurons.

Implements [SaliencyMap](#).

Definition at line 231 of file SaliencyMap.C.

References [Image](#)< T >::begin(), [Image](#)< [LeakyIntegrator](#) >::beginw(), [Image](#)< T >::end(), [Image](#)< [LeakyIntegrator](#) >::endw(), [Image](#)< T >::getDims(), [ModelParam](#)< float >::getVal(), [Image](#)< [LeakyIntegrator](#) >::initialized(), and [Image](#)< [LeakyIntegrator](#) >::resize().

### 10.245.3.11 void SaliencyMapStd::reset () [virtual]

Reset to initial state just after construction.

Implements [SaliencyMap](#).

Definition at line 372 of file SaliencyMap.C.

References [Image](#)< [LeakyIntegrator](#) >::freeMem().

### 10.245.3.12 void SaliencyMapStd::saccadicSuppression (const bool on) [virtual]

Turn saccadic suppression on/off.

The behavior here is that turning saccadic suppression on will open the inhibitory conductances of the SM neurons just like inhibition of return does. This function should be called repeatedly during suppression as, otherwise, the inhibitory conductances have a natural tendency to fade away (as is needed for IOR to pregressively fade away). When this function is called to turn saccadic suppression off, the inhibitory conductances are violently reset to zero. So, for full saccadic suppression, call SM.saccadicSuppression(true) at every time step during the suppression time interval, then call SM.saccadicSuppression(false) once at the end of the suppression period. Calling SM.saccadicSuppression(true) less often (or just once at the start of a saccade) will yield milder suppression.

Implements [SaliencyMap](#).

Definition at line 307 of file SaliencyMap.C.

References [Image](#)< [LeakyIntegrator](#) >::beginw(), [Image](#)< [LeakyIntegrator](#) >::endw(), and [ModelParam](#)< bool >::getVal().

## 10.245.4 Member Data Documentation

### 10.245.4.1 [ModelParam<float>](#) [SaliencyMapStd::itsGinhDecay](#) [protected]

decay for IOR inhibition

Definition at line 332 of file SaliencyMap.H.

The documentation for this class was generated from the following files:

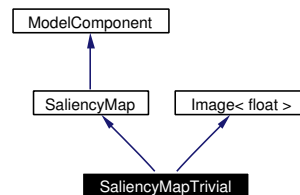
- [SaliencyMap.H](#)
- [SaliencyMap.C](#)

## 10.246 SaliencyMapTrivial Class Reference

The trivial saliency map: a 2D array of floats.

```
#include <SaliencyMap.H>
```

Inheritance diagram for SaliencyMapTrivial:



Collaboration diagram for SaliencyMapTrivial:

### Public Member Functions

- [SaliencyMapTrivial](#) ([ModelManager](#) &mgr, const [std::string](#) &descr-Name="Saliency Map Trivial", const [std::string](#) &tagName="Saliency-MapTrivial")  
*Initialized constructor.*
- virtual [~SaliencyMapTrivial](#) ()  
*Destructor.*
- virtual void [reset](#) ()  
*Reset to initial state just after construction.*
- virtual void [input](#) (const [Image< float >](#) &in)  
*Set new input currents for all neurons.*
- virtual [Image< float >](#) [getV](#) (const bool normalize=false) const  
*Return all our membrane potential voltages as an [Image<float>](#).*
- virtual float [getV](#) (const [Point2D](#) &p) const  
*Access function using [Point2D](#).*
- virtual int [getWidth](#) () const  
*Get width of saliency map.*

- virtual int [getHeight](#) () const  
*Get height of saliency map.*
- virtual void [evolve](#) (const double t)  
*Integrate inputs until time t (in s) and update membrane potential.*
- virtual void [depress](#) (const [Point2D](#) &winner, const double &pampl, const double &mampl, const double &psdev, const double &msdev)  
*difference-of-Gaussians inhibition of return*
- virtual void [depress](#) (const [Point2D](#) &winner, const double &ampl, const [Image](#)< byte > &objectMask)  
*object-based inhibition of return*
- virtual void [saccadicSuppression](#) (const bool on)  
*Turn saccadic suppression on/off.*
- virtual void [blinkSuppression](#) (const bool on)  
*Turn blink suppression on/off.*

### 10.246.1 Detailed Description

The trivial saliency map: a 2D array of floats.

This is a 2D salience map. It is an [Image](#)<float> and inherits all the standard methods of class [Image](#). To avoid confusion, we here add explicit [input\(\)](#) and [getV\(\)](#) methods. [evolve\(\)](#) does nothing in this version.

Definition at line 342 of file SaliencyMap.H.

### 10.246.2 Constructor & Destructor Documentation

#### 10.246.2.1 SaliencyMapTrivial::SaliencyMapTrivial ([ModelManager](#) & mgr, const [std::string](#) & descrName = "Saliency Map Trivial", const [std::string](#) & tagName = "SaliencyMapTrivial")

Initialized constructor.

The SM will be resized and initialized the first time [input\(\)](#) is called

Definition at line 390 of file SaliencyMap.C.



### 10.246.2.2 SaliencyMapTrivial::~~SaliencyMapTrivial () [virtual]

Destructor.

Definition at line 397 of file SaliencyMap.C.

## 10.246.3 Member Function Documentation

### 10.246.3.1 void SaliencyMapTrivial::blinkSuppression (const bool on) [virtual]

Turn blink suppression on/off.

Implements [SaliencyMap](#).

Definition at line 461 of file SaliencyMap.C.

References [ModelParam< bool >::getVal\(\)](#).

### 10.246.3.2 void SaliencyMapTrivial::depress (const [Point2D](#) & winner, const double & ampl, const [Image](#)< byte > & objectMask) [virtual]

object-based inhibition of return

The object mask defines where in the image the inhibition will be applied. The winner coordinates and ampl are used to calibrate the inhibition strength

Implements [SaliencyMap](#).

Definition at line 439 of file SaliencyMap.C.

References [Image< float >::rectify\(\)](#).

### 10.246.3.3 void SaliencyMapTrivial::depress (const [Point2D](#) & winner, const double & pamp, const double & mampl, const double & psdev, const double & msdev) [virtual]

difference-of-Gaussians inhibition of return

The positive Gaussian is defined by its amplitude pamp and width psdev; the negative Gaussian by mampl and msdev. Both centered at winner.

Implements [SaliencyMap](#).

Definition at line 418 of file SaliencyMap.C.

References [Image< float >::beginw\(\)](#), [getHeight\(\)](#), [getWidth\(\)](#), [Point2D::i](#), [Point2D::j](#), and [SaliencyMap::w\(\)](#).

**10.246.3.4 void SaliencyMapTrivial::evolve (const double t) [virtual]**

Integrate inputs until time t (in s) and update membrane potential.

Implements [SaliencyMap](#).

Definition at line 475 of file SaliencyMap.C.

**10.246.3.5 int SaliencyMapTrivial::getHeight () const [virtual]**

Get height of saliency map.

Implements [SaliencyMap](#).

Definition at line 491 of file SaliencyMap.C.

References [Image< T >::getHeight\(\)](#).

Referenced by [depress\(\)](#).

**10.246.3.6 float SaliencyMapTrivial::getV (const [Point2D](#) & p) const [virtual]**

Access function using [Point2D](#).

Implements [SaliencyMap](#).

Definition at line 479 of file SaliencyMap.C.

References [Image< float >::getVal\(\)](#).

**10.246.3.7 [Image< float >](#) SaliencyMapTrivial::getV (const bool *normalize* = false) const [virtual]**

Return all our membrane potential voltages as an [Image<float>](#).

Implements [SaliencyMap](#).

Definition at line 407 of file SaliencyMap.C.

References [ModelParam< float >::getVal\(\)](#), and [Image< T >::normalize\(\)](#).

**10.246.3.8 int SaliencyMapTrivial::getWidth () const [virtual]**

Get width of saliency map.

Implements [SaliencyMap](#).

Definition at line 487 of file SaliencyMap.C.

References [Image< T >::getWidth\(\)](#).

Referenced by `depress()`.

#### 10.246.3.9 `void SaliencyMapTrivial::input (const Image< float > & in)` [virtual]

Set new input currents for all neurons.

Implements [SaliencyMap](#).

Definition at line 401 of file `SaliencyMap.C`.

#### 10.246.3.10 `void SaliencyMapTrivial::reset ()` [virtual]

Reset to initial state just after construction.

Implements [SaliencyMap](#).

Definition at line 483 of file `SaliencyMap.C`.

References `Image< float >::freeMem()`.

#### 10.246.3.11 `void SaliencyMapTrivial::saccadicSuppression (const bool on)` [virtual]

Turn saccadic suppression on/off.

The behavior here is that turning saccadic suppression on will open the inhibitory conductances of the SM neurons just like inhibition of return does. This function should be called repeatedly during suppression as, otherwise, the inhibitory conductances have a natural tendency to fade away (as is needed for IOR to pregressively fade away). When this function is called to turn saccadic suppression off, the inhibitory conductances are violently reset to zero. So, for full saccadic suppression, call `SM.saccadicSuppression(true)` at every time step during the suppression time interval, then call `SM.saccadicSuppression(false)` once at the end of the suppression period. Calling `SM.saccadicSuppression(true)` less often (or just once at the start of a saccade) will yield milder suppression.

Implements [SaliencyMap](#).

Definition at line 448 of file `SaliencyMap.C`.

References `ModelParam< bool >::getVal()`.

The documentation for this class was generated from the following files:

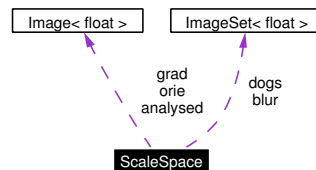
- [SaliencyMap.H](#)
- [SaliencyMap.C](#)

## 10.247 ScaleSpace Class Reference

ScaleSpace.

```
#include <ScaleSpace.H>
```

Collaboration diagram for ScaleSpace:



### Public Member Functions

- [ScaleSpace \(\)](#)  
*Constructor.*
- [~ScaleSpace \(void\)](#)  
*Destructor.*
- void [reset \(\)](#)  
*reset*
- void [initScaleSpace](#) (float currentscale, const [Image< float >](#) &image)  
*Init this ScaleSpace provided the scale dimension and the picture.*
- int [findKeypoints](#) ([VisualObject](#) &obj)  
*Find the keypoints in the previously provided picture.*

### 10.247.1 Detailed Description

ScaleSpace.

It handles a scale space with all the pictures associated. Implements what is needed for sift detection in one level of the full SpaceScale

Definition at line 60 of file ScaleSpace.H.

## 10.247.2 Constructor & Destructor Documentation

### 10.247.2.1 ScaleSpace::ScaleSpace () [inline]

Constructor.

Definition at line 64 of file ScaleSpace.H.

### 10.247.2.2 ScaleSpace::~ScaleSpace (void) [inline]

Destructor.

Definition at line 67 of file ScaleSpace.H.

References `reset()`.

## 10.247.3 Member Function Documentation

### 10.247.3.1 int ScaleSpace::findKeypoints (VisualObject & obj)

Find the keypoints in the previously provided picture.

Returns the number of keypoints found

Definition at line 125 of file ScaleSpace.C.

Referenced by `InferoTemporal::computeVisualObject()`.

### 10.247.3.2 void ScaleSpace::initScaleSpace (float currentscale, const Image< float > & image)

Init this ScaleSpace provided the scale dimension and the picture.

Definition at line 49 of file ScaleSpace.C.

References `convGauss()`, `Image< PixRGB< float > >::getHeight()`, `Image< PixRGB< float > >::getWidth()`, `image`, `reset()`, `sqrt()`, and `ZEROS`.

Referenced by `InferoTemporal::computeVisualObject()`.

### 10.247.3.3 void ScaleSpace::reset () [inline]

reset

Definition at line 70 of file ScaleSpace.H.

References `ImageSet< float >::clear()`.

Referenced by `initScaleSpace()`, and `~ScaleSpace()`.

The documentation for this class was generated from the following files:

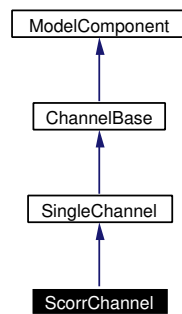
- [ScaleSpace.H](#)
- [ScaleSpace.C](#)

## 10.248 ScorrChannel Class Reference

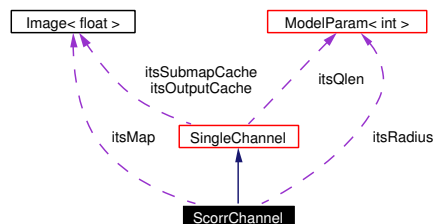
A spatial correlation channel.

```
#include <Channels.H>
```

Inheritance diagram for ScorrChannel:



Collaboration diagram for ScorrChannel:



### Public Member Functions

- `ScorrChannel (ModelManager &mgr)`  
*Construct with standard params.*
- virtual void `exportOptions` (const int whatToExport, const bool recurse=true)  
*Export some of our `ModelParam` parameters as command-line-options.*
- virtual `~ScorrChannel ()`  
*Destructor.*
- virtual bool `outputAvailable ()` const

*overload*

## Protected Member Functions

- virtual void [doInput](#) (const [Image](#)< [PixRGB](#)< float > > \*, const [Image](#)< float > \*bwimg, const double t, const [Image](#)< byte > &clipMask)

*special input function*

- virtual [Image](#)< float > [getOutput](#) ()

*special output function*

## Protected Attributes

- [ModelParam](#)< int > [itsRadius](#)

*Radius (in saliency map pixels).*

- [Image](#)< float > [itsMap](#)

*our temporal correlation map*

### 10.248.1 Detailed Description

A spatial correlation channel.

Spatial correlation is computed at every location between the local patch and other patches at a given radius from the local patch. Radius is given in saliency map pixels. Note that in this channel, high 'saliency' means low spatial correlation, i.e., lots of change. CAUTION: this channel will throw a fatal exception if your [LevelSpec](#) contains more than one level or if that level is different from the maplevel or if the deltas are non-zero.

Definition at line 578 of file Channels.H.

### 10.248.2 Constructor & Destructor Documentation

#### 10.248.2.1 ScorrChannel::ScorrChannel ([ModelManager](#) & mgr)

Construct with standard params.

Definition at line 2745 of file Channels.C.

References SCORR.



**10.248.2.2 ScorrChannel::~ScorrChannel () [virtual]**

Destructor.

Definition at line 2764 of file Channels.C.

**10.248.3 Member Function Documentation****10.248.3.1 void ScorrChannel::doInput (const [Image](#)< [PixRGB](#)< float > > \*, const [Image](#)< float > \* *bwimg*, const double *t*, const [Image](#)< byte > & *clipMask*) [protected, virtual]**

special input function

Reimplemented from [SingleChannel](#).

Definition at line 2778 of file Channels.C.

References [ASSERT](#), [Image](#)< float >::beginw(), [Image](#)< T >::clear(), [LevelSpec](#)::delMax(), [LevelSpec](#)::delMin(), [Image](#)< T >::getHeight(), [ModelParam](#)< int >::getVal(), [ModelParam](#)< [LevelSpec](#) >::getVal(), [Image](#)< T >::getWidth(), [Point2D](#)::i, [itsRadius](#), [Point2D](#)::j, [LevelSpec](#)::levMax(), [LevelSpec](#)::levMin(), [LevelSpec](#)::mapLevel(), [NO\\_INIT](#), [Image](#)< float >::resize(), [SQR](#)(), and [uint](#).

**10.248.3.2 void ScorrChannel::exportOptions (const int *whatToExport*, const bool *recurse* = true) [virtual]**

Export some of our [ModelParam](#) parameters as command-line-options.

See the base function in [ModelComponent.H](#) for info.

Reimplemented from [SingleChannel](#).

Definition at line 2753 of file Channels.C.

References [SingleChannel](#)::exportOptions(), [itsRadius](#), [OPTEXP\\_CORE](#), and [ModelManager](#)::requestOption().

**10.248.3.3 [Image](#)< float > ScorrChannel::getOutput () [protected, virtual]**

special output function

Reimplemented from [SingleChannel](#).

Definition at line 2842 of file Channels.C.

**10.248.3.4** `bool ScorrChannel::outputAvailable () const` [virtual]

overload

Reimplemented from [SingleChannel](#).

Definition at line 2768 of file Channels.C.

References `Image< float >::initialized()`.

**10.248.4 Member Data Documentation****10.248.4.1** `Image<float> ScorrChannel::itsMap` [protected]

our temporal correlation map

Definition at line 606 of file Channels.H.

**10.248.4.2** `ModelParam<int> ScorrChannel::itsRadius` [protected]

Radius (in saliency map pixels).

Definition at line 596 of file Channels.H.

Referenced by `doInput()`, and `exportOptions()`.

The documentation for this class was generated from the following files:

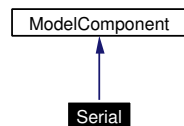
- [Channels.H](#)
- [Channels.C](#)

## 10.249 Serial Class Reference

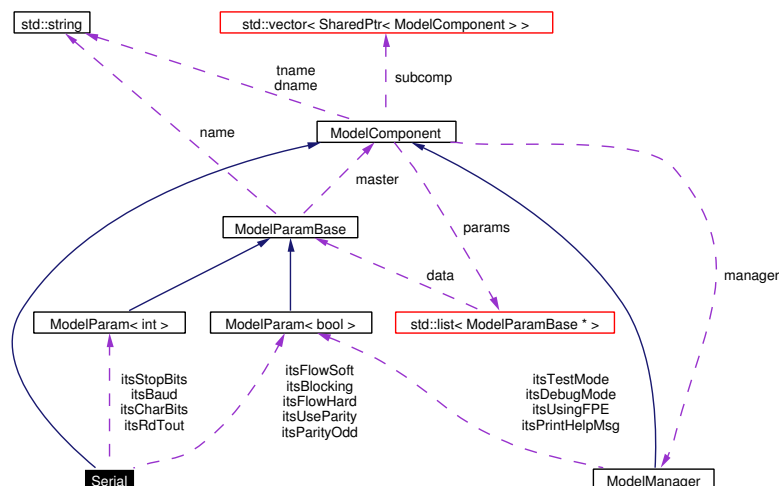
Interface to a serial port.

```
#include <serial.H>
```

Inheritance diagram for Serial:



Collaboration diagram for Serial:



### Public Member Functions

- [Serial](#) ([ModelManager](#) &mgr, const `std::string` &descrName="Serial Port", const `std::string` &tagName="SerialPort")

*Constructor.*

- [~Serial](#) (void)

*destructor*

- void [configure](#) (const char \*dev="/dev/ttyS0", const int speed=9600, const char \*format="8N1", const bool flowSoft=false, const bool flowHard=true, const int=0)  
*Configure the port before it is started.*
- [serialError setSpeed](#) (const int speed)  
*Set serial port speed for both input and output.*
- [serialError setFlowControl](#) (const bool useHard, const bool useSoft)  
*Set the serial port flow control on Input as well as Output.*
- [serialError setCharBits](#) (const int bits)  
*Set character size.*
- [serialError setParity](#) (const bool useParity, const bool oddParity)  
*Set parity mode.*
- [serialError setStopBits](#) (const int bits)  
*Set number of stop bits.*
- [serialError setBlocking](#) (const bool [blocking](#))  
*Set the access to blocking or not.*
- void [toggleDTR](#) (const time\_t millisec)  
*Set the DTR mode off momentarily.*
- void [sendBreak](#) (void)  
*transmit continuous stream of zero-valued bits for specific duration.*
- int [read](#) (char \*buffer, const int nbytes)  
*attempt to read up to nbytes from serial port into the buffer*
- char [read](#) (void)  
*attempts to get the next character from the serial port*
- int [write](#) (const char \*buffer, const int nbytes)  
*write bytes to the port*
- void [perror](#) (void)  
*print a verbal message indicating the last error.*

## Protected Member Functions

- void [start1](#) ()  
*open the port and get started*
- void [stop2](#) ()  
*close the port and get stopped*

## Protected Attributes

- [ModelParam](#)< [std::string](#) > [itsDevName](#)
- [ModelParam](#)< int > [itsBaud](#)  
*Baudrate.*
- [ModelParam](#)< int > [itsCharBits](#)  
*number of useful bits per char*
- [ModelParam](#)< int > [itsStopBits](#)  
*number of stop bits*
- [ModelParam](#)< bool > [itsUseParity](#)  
*use parity if true*
- [ModelParam](#)< bool > [itsParityOdd](#)  
*parity is odd if true, otherwise even*
- [ModelParam](#)< bool > [itsFlowHard](#)  
*use hardware flow control if true*
- [ModelParam](#)< bool > [itsFlowSoft](#)  
*use software flow control if true*
- [ModelParam](#)< int > [itsRdTout](#)  
*read timeout in 1/10000s*
- [ModelParam](#)< bool > [itsBlocking](#)  
*use blocking mode*

### 10.249.1 Detailed Description

Interface to a serial port.

The port will be open at [start\(\)](#) time; see [ModelComponent.H](#) for details. Because typically one may use several serial ports with different configs, this [ModelComponent](#) does not export any command-line option (there would be too many otherwise if many ports are used). Typically, thus, the port can be configured either via config files or using the various access functions provided in this class.

Definition at line 81 of file [serial.H](#).

### 10.249.2 Constructor & Destructor Documentation

#### 10.249.2.1 [Serial::Serial](#) ([ModelManager](#) & mgr, const [std::string](#) & descrName = "Serial Port", const [std::string](#) & tagName = "SerialPort")

Constructor.

Definition at line 43 of file [serial.C](#).

#### 10.249.2.2 [Serial::~Serial](#) (void)

destructor

Definition at line 545 of file [serial.C](#).

### 10.249.3 Member Function Documentation

#### 10.249.3.1 void [Serial::configure](#) (const char \* dev = "/dev/ttyS0", const int speed = 9600, const char \* format = "8N1", const bool flowSoft = false, const bool flowHard = true, const int = 0)

Configure the port before it is started.

This will update our internal [ModelParam](#) values to the specified ones; thus this should be called before [start\(\)](#). In contrast, the setXXX functions below are for reconfiguration at runtime, once the port is already open and running. This function is provided as a shortcut to set a bunch of [ModelParam](#) values in one simple call.

##### Parameters:

*dev* device name (e.g., "/dev/ttyS0")

*speed* speed in bauds. See [setSpeed\(\)](#) for valid values

*format* a 3-char string for charbits, parity, and stop bits; for example "8N1"; parity should be "N", "E" or "O".

***flowSoft*** use software flow control if true  
***flowHard*** use hardware flow control if true  
***tout*** read timeout in 1/10000s or 0 for no timeout

Definition at line 92 of file serial.C.

References itsBaud, itsCharBits, itsFlowHard, itsFlowSoft, itsParityOdd, itsRdTout, itsUseParity, ModelParam< bool >::setVal(), and ModelParam< int >::setVal().

### 10.249.3.2 void Serial::perror (void)

print a verbal message indicating the last error.

Definition at line 486 of file serial.C.

Referenced by start1().

### 10.249.3.3 char Serial::read (void)

attempts to get the next character from the serial port

for comptability with dirk's code

Definition at line 452 of file serial.C.

### 10.249.3.4 int Serial::read (char \* *buffer*, const int *nbytes*)

attempt to read up to nbytes from serial port into the buffer

#### Parameters:

***buffer*** holds bytes after read number of bytes to attempt to read

#### Returns:

0 on a timeout, -1 on error, number of bytes actually read on a success

Definition at line 437 of file serial.C.

### 10.249.3.5 void Serial::sendBreak (void)

transmit continuous stream of zero-valued bits for specific duration.

Definition at line 355 of file serial.C.

Referenced by stop2().

**10.249.3.6 [serialError](#) Serial::setBlocking (const bool *blocking*)**

Set the access to blocking or not.

If blocking, a write will block until all data has actually been sent through; otherwise it will return immediately while the data is being transmitted.

Definition at line 319 of file serial.C.

References itsBlocking, serialError, and ModelParam< bool >::setVal().

Referenced by start1().

**10.249.3.7 [serialError](#) Serial::setCharBits (const int *bits*)**

Set character size.

This function is to change the char size while already started.

**Returns:**

0 on success.

**Parameters:**

***bits*** character size to use; valid values are 5, 6, 7, 8.

Definition at line 231 of file serial.C.

References itsCharBits, serialError, and ModelParam< int >::setVal().

Referenced by start1().

**10.249.3.8 [serialError](#) Serial::setFlowControl (const bool *useHard*, const bool *useSoft*)**

Set the serial port flow control on Input as well as Output.

This function is to change the flow control while already started.

**Parameters:**

***useHard*** use hardware flow control if true

***useSoft*** use software flow control if true

**Returns:**

0 or serialErrSuccess on success

Definition at line 203 of file serial.C.

References itsFlowHard, itsFlowSoft, serialError, and ModelParam< bool >::setVal().

Referenced by start1().



### 10.249.3.9 **serialError** Serial::setParity (const bool *useParity*, const bool *oddParity*)

Set parity mode.

This function is to change the parity while already started.

**Parameters:**

*useParity* use some parity if true

*oddParity* parity is odd if true, otherwise even

**Returns:**

0 on success.

Definition at line 263 of file serial.C.

References itsParityOdd, itsUseParity, serialError, and ModelParam< bool >::setVal().

Referenced by start1().

### 10.249.3.10 **serialError** Serial::setSpeed (const int *speed*)

Set serial port speed for both input and output.

This function is to change the speed while already started.

**Parameters:**

*speed* to select. 0 signifies modem "hang up" and possible values are 0, 110, 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200.

**Returns:**

0 or serialErrSuccess on success. The [perror\(\)](#) method can be used to print a longer description of the error.

Definition at line 127 of file serial.C.

References itsBaud, serialError, and ModelParam< int >::setVal().

Referenced by start1().

### 10.249.3.11 **serialError** Serial::setStopBits (const int *bits*)

Set number of stop bits.

This function is to change the stop bits while started.

**Returns:**

0 on success.

**Parameters:**

*bits* stop bits (only values 1 and 2 are valid).

Definition at line 294 of file serial.C.

References serialError, and ModelParam< int >::setVal().

**10.249.3.12 void Serial::start1 () [protected, virtual]**

open the port and get started

Reimplemented from [ModelComponent](#).

Definition at line 60 of file serial.C.

References ModelParam< bool >::getVal(), ModelParam< int >::getVal(), itsBaud, itsBlocking, itsCharBits, itsFlowHard, itsFlowSoft, itsParityOdd, itsUseParity, per-ror(), setBlocking(), setCharBits(), setFlowControl(), setParity(), and setSpeed().

**10.249.3.13 void Serial::stop2 () [protected, virtual]**

close the port and get stopped

Reimplemented from [ModelComponent](#).

Definition at line 86 of file serial.C.

References sendBreak().

**10.249.3.14 void Serial::toggleDTR (const time\_t millisec)**

Set the DTR mode off momentarily.

**Parameters:**

*millisec* number of milliseconds.

Definition at line 332 of file serial.C.

**10.249.3.15 int Serial::write (const char \* buffer, const int nbytes)**

write bytes to the port

**Parameters:**

*buffer* begin writing from the location buffer.

*nbytes* number of bytes to write

**Returns:**

-1 on failure, number of bytes written on success

Definition at line 463 of file serial.C.

## 10.249.4 Member Data Documentation

### 10.249.4.1 **ModelParam**<int> **Serial::itsBaud** [protected]

Baudrate.

Definition at line 179 of file serial.H.

Referenced by `configure()`, `setSpeed()`, and `start1()`.

### 10.249.4.2 **ModelParam**<bool> **Serial::itsBlocking** [protected]

use blocking mode

Definition at line 187 of file serial.H.

Referenced by `setBlocking()`, and `start1()`.

### 10.249.4.3 **ModelParam**<int> **Serial::itsCharBits** [protected]

number of useful bits per char

Definition at line 180 of file serial.H.

Referenced by `configure()`, `setCharBits()`, and `start1()`.

### 10.249.4.4 **ModelParam**<bool> **Serial::itsFlowHard** [protected]

use hardware flow control if true

Definition at line 184 of file serial.H.

Referenced by `configure()`, `setFlowControl()`, and `start1()`.

### 10.249.4.5 **ModelParam**<bool> **Serial::itsFlowSoft** [protected]

use software flow control if true

Definition at line 185 of file serial.H.

Referenced by `configure()`, `setFlowControl()`, and `start1()`.

**10.249.4.6** [ModelParam<bool>](#) [Serial::itsParityOdd](#) [protected]

parity is odd if true, otherwise even

Definition at line 183 of file serial.H.

Referenced by [configure\(\)](#), [setParity\(\)](#), and [start1\(\)](#).

**10.249.4.7** [ModelParam<int>](#) [Serial::itsRdTout](#) [protected]

read timeout in 1/10000s

Definition at line 186 of file serial.H.

Referenced by [configure\(\)](#).

**10.249.4.8** [ModelParam<int>](#) [Serial::itsStopBits](#) [protected]

number of stop bits

Definition at line 181 of file serial.H.

**10.249.4.9** [ModelParam<bool>](#) [Serial::itsUseParity](#) [protected]

use parity if true

Definition at line 182 of file serial.H.

Referenced by [configure\(\)](#), [setParity\(\)](#), and [start1\(\)](#).

The documentation for this class was generated from the following files:

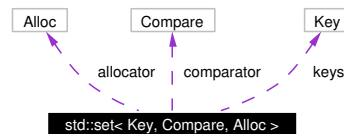
- [serial.H](#)
- [serial.C](#)

## 10.250 `std::set< Key, Compare, Alloc >` Class Template Reference

The STL set container class. Use with `#include` .

```
#include <dummySTL.H>
```

Collaboration diagram for `std::set< Key, Compare, Alloc >`:



### Public Attributes

- Key **keys**
- Compare **comparator**
- Alloc **allocator**

#### 10.250.1 Detailed Description

```
template<class Key, class Compare, class Alloc> class std::set< Key, Compare, Alloc >
```

The STL set container class. Use with `#include` .

See [SGI's documentation](#) of `std::set`.

Definition at line 37 of file `dummySTL.H`.

The documentation for this class was generated from the following file:

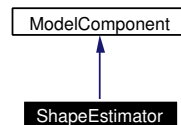
- [dummySTL.H](#)

## 10.251 ShapeEstimator Class Reference

everything realted to the shape estimation procedure

```
#include <ShapeEstimator.H>
```

Inheritance diagram for ShapeEstimator:



Collaboration diagram for ShapeEstimator:

### Public Member Functions

- [ShapeEstimator](#) ([ModelManager](#) &mgr, const [std::string](#) &descrName="Shape Estimator", const [std::string](#) &tagName="shapeestimator", const [SharedPtr](#)<[VisualCortex](#) > vcx=[SharedPtr](#)<[VisualCortex](#) >(NULL))

*Constructor.*

- void [setVC](#) (const [SharedPtr](#)<[VisualCortex](#) > &vc)  
*set the [VisualCortex](#) if it was not given at construction*
- virtual void [exportOptions](#) (const int whatToExport, const bool recurse=true)  
*Export some of our [ModelParam](#) parameters as command-line-options.*
- virtual void [reset](#) (const bool recurse=true)  
*Reset ShapeEstimator.*
- void [compute](#) (const [Point2D](#) &winner)  
*re-compute for a new winner*
- [Point2D](#) [getWinner](#) () const  
*return the winner from the last compute call*
- [ChannelBase](#) & [getChannel](#) () const  
*returns a pointer to the winning channel*
- [SingleChannel](#) & [getSingleChannel](#) () const  
*returns pointer to the winning [SingleChannel](#) (only for mode = featureMap)*

- const [ImageSet](#)< float > & [getPyramid](#) (const int idx=0) const  
*returns pyramid of the winning [SingleChannel](#) (only for mode = featureMap)*
- int [getSubmapNum](#) () const  
*returns the number (the index) of the winning submap in the winning channel*
- [Image](#)< float > & [getWinningMap](#) ()  
*returns the winning map (in saliency map coordinates)*
- [std::string](#) [getWinningLabel](#) () const  
*the label of the winning map*
- bool [isInitialized](#) () const  
*whether compute has been run at least once*
- [Image](#)< byte > & [getObjectMask](#) ()  
*return a binary mask of the attended object {0,255} in saliency map coordinates*
- int [getObjectArea](#) ()  
*returns the area of the object Mask, scaled up to image coordinates*
- [Image](#)< [PixRGB](#)< byte > > [getObjectImage](#) ([Image](#)< [PixRGB](#)< byte > > &scene)  
*returns the object image*
- [Image](#)< byte > & [getIORMask](#) ()  
*return a graded mask for IOR [0,255] in saliency map coordinates*
- [Image](#)< float > & [getSmoothMask](#) ()  
*return an upscaled and smoothed mask [0.0,1.0] in image coordinates*
- [Image](#)< float > [getCumSmoothMask](#) () const  
*return the cumulative smoothed mask [0.0,1.0] in image coordinates*
- [Image](#)< float > [getNegCumSmoothMask](#) () const  
*return the negative cumulative smoothed mask [0.0,1.0] in image coordinates*
- void [resetCumMask](#) ()  
*reset the cumulative mask*

## Protected Attributes

- [ModelParam< ShapeEstimatorMode > itsMode](#)  
*Determines what information is used as a source for the shape estimation.*
- [ModelParam< ShapeEstimatorSmoothMethod > itsSmMethod](#)  
*Determines the Smoothing Method used to extract a map in image coordinates.*

### 10.251.1 Detailed Description

everything related to the shape estimation procedure

class to take care of all things related to the shape estimation procedure published in Walther et al, BMCV 02

Definition at line 58 of file ShapeEstimator.H.

### 10.251.2 Constructor & Destructor Documentation

**10.251.2.1** `ShapeEstimator::ShapeEstimator (ModelManager & mgr, const std::string & descrName = "Shape Estimator", const std::string & tagName = "shapeestimator", const SharedPtr< VisualCortex > vcx = SharedPtr< VisualCortex >(NULL))`

Constructor.

#### Parameters:

`vcx` this is the hook for the class to obtain its information on channels, submaps etc.

Definition at line 55 of file ShapeEstimator.C.

References `SEMfeatureMap`, `SESMgaussian`, `ShapeEstimatorMode`, and `ShapeEstimatorSmoothMethod`.

### 10.251.3 Member Function Documentation

**10.251.3.1** `void ShapeEstimator::compute (const Point2D & winner)`

re-compute for a new winner

Definition at line 109 of file ShapeEstimator.C.

References `ASSERT`, `Image< byte >::freeMem()`, `Image< float >::freeMem()`, `SharedPtr< VisualCortex >::get()`, `Image< T >::getDims()`, `Image< T >::getVal()`,



ModelParam< ShapeEstimatorMode >::getVal(), itsMode, SharedPtr< ChannelBase >::reset(), SEMconspicuityMap, SEMsaliencyMap, and uint.

#### 10.251.3.2 void ShapeEstimator::exportOptions (const int *whatToExport*, const bool *recurse* = true) [virtual]

Export some of our [ModelParam](#) parameters as command-line-options.

See the base function in [ModelComponent.H](#) for info.

Reimplemented from [ModelComponent](#).

Definition at line 79 of file ShapeEstimator.C.

References [ModelComponent::exportOptions\(\)](#), [itsMode](#), [itsSmMethod](#), [OPTEXP\\_CORE](#), and [ModelManager::requestOption\(\)](#).

#### 10.251.3.3 [ChannelBase](#) & ShapeEstimator::getChannel () const

returns a pointer to the winning channel

for mode = saliencyMap the pointer will be empty

Definition at line 462 of file ShapeEstimator.C.

References [ASSERT](#).

#### 10.251.3.4 [Image](#)< float > ShapeEstimator::getCumSmoothMask () const

return the cumulative smoothed mask [0.0,1.0] in image coordinates

IMPORTANT: for this to work, getSmoothMask has to be called at least once after each new compute call

Definition at line 434 of file ShapeEstimator.C.

References [Image< T >::clamp\(\)](#), and [Image< float >::initialized\(\)](#).

#### 10.251.3.5 [Image](#)< byte > & ShapeEstimator::getIORMask ()

return a graded mask for IOR [0,255] in saliency map coordinates

Definition at line 352 of file ShapeEstimator.C.

References [ASSERT](#), [Image< byte >::freeMem\(\)](#), [getObjectMask\(\)](#), [ModelParam< ShapeEstimatorMode >::getVal\(\)](#), [Image< byte >::initialized\(\)](#), and [itsMode](#).

**10.251.3.6 [Image](#)< float > ShapeEstimator::getNegCumSmoothMask () const**

return the negative cumulative smoothed mask [0.0,1.0] in image coordinates

IMPORTANT: for this to work, getSmoothMask has to be called at least once after each new compute call

Definition at line 444 of file ShapeEstimator.C.

References [Image](#)< T >::clamp(), and [Image](#)< float >::initialized().

**10.251.3.7 int ShapeEstimator::getObjectArea ()**

returns the area of the object Mask, scaled up to image coordinates

Definition at line 250 of file ShapeEstimator.C.

References ASSERT, getObjectMask(), [Image](#)< byte >::getSize(), [Image](#)< byte >::initialized(), and sum().

**10.251.3.8 [Image](#)< [PixRGB](#)< byte > > ShapeEstimator::getObjectImage ([Image](#)< [PixRGB](#)< byte > > & scene)**

returns the object image

Definition at line 266 of file ShapeEstimator.C.

References crop(), [Image](#)< float >::getHeight(), getSmoothMask(), [Image](#)< float >::getVal(), [Image](#)< float >::getWidth(), and [Image](#)< float >::initialized().

**10.251.3.9 [Image](#)< byte > & ShapeEstimator::getObjectMask ()**

return a binary mask of the attended object {0,255} in saliency map coordinates

Definition at line 200 of file ShapeEstimator.C.

References ASSERT, [Point2D](#)::fitToDims(), [Image](#)< byte >::freeMem(), [ModelParam](#)< [ShapeEstimatorMode](#) >::getVal(), itsMode, [Image](#)< float >::normalize(), segmentObjectClean(), and uint.

Referenced by getIORMask(), getObjectArea(), and getSmoothMask().

**10.251.3.10 const [ImageSet](#)< float > & ShapeEstimator::getPyramid (const int idx = 0) const**

returns pyramid of the winning [SingleChannel](#) (only for mode = featureMap)

**Parameters:**

**idx** index of the pyramid in the queue (default: 0)

Definition at line 486 of file ShapeEstimator.C.

References ASSERT, getSingleChannel(), ModelParam< ShapeEstimatorMode >::getVal(), itsMode, SingleChannel::pyramid(), and SEMfeatureMap.

**10.251.3.11 [SingleChannel](#) & ShapeEstimator::getSingleChannel () const**

returns pointer to the winning [SingleChannel](#) (only for mode = featureMap)

Definition at line 466 of file ShapeEstimator.C.

References ASSERT, SharedPtr< ChannelBase >::get(), ModelParam< ShapeEstimatorMode >::getVal(), itsMode, ChannelBase::numChans(), SEMfeatureMap, ComplexChannel::subChanFromSubmapNum(), and uint.

Referenced by getPyramid().

**10.251.3.12 [Image< float >](#) & ShapeEstimator::getSmoothMask ()**

return an upscaled and smoothed mask [0.0,1.0] in image coordinates

Definition at line 368 of file ShapeEstimator.C.

References ASSERT, binaryReverse(), chamfer34(), Image< float >::clamp(), Image< byte >::drawDisk(), Image< float >::freeMem(), getObjectMask(), ModelParam< ShapeEstimatorSmoothMethod >::getVal(), ModelParam< ShapeEstimatorMode >::getVal(), Image< byte >::initialized(), Image< float >::initialized(), itsMode, itsSmMethod, Image< float >::normalize(), openImg(), scaleBlock(), SESMchamfer, SESMgaussian, takeMax(), and ZEROS.

Referenced by getObjectImage().

**10.251.3.13 int ShapeEstimator::getSubmapNum () const**

returns the number (the index) of the winning submap in the winning channel

makes only sense for mode = featureMap

Definition at line 493 of file ShapeEstimator.C.

References ASSERT.

**10.251.3.14 [Point2D](#) ShapeEstimator::getWinner () const**

return the winner from the last compute call

Definition at line 458 of file ShapeEstimator.C.

References ASSERT.

#### 10.251.3.15 `std::string ShapeEstimator::getWinningLabel () const`

the label of the winning map

Definition at line 501 of file ShapeEstimator.C.

References ASSERT.

#### 10.251.3.16 `Image< float > & ShapeEstimator::getWinningMap ()`

returns the winning map (in saliency map coordinates)

for mode == saliencyMap, a copy of the saliency map; for mode == conspicuityMap, the output of the winning channel; for mode == feature-Map, winning submap of the winning channel

Definition at line 497 of file ShapeEstimator.C.

References ASSERT.

#### 10.251.3.17 `bool ShapeEstimator::isInitialized () const`

whether compute has been run at least once

Definition at line 505 of file ShapeEstimator.C.

#### 10.251.3.18 `void ShapeEstimator::reset (const bool recurse = true) [virtual]`

Reset ShapeEstimator.

See the base function in [ModelComponent.H](#) for info.

Reimplemented from [ModelComponent](#).

Definition at line 91 of file ShapeEstimator.C.

References `Image< byte >::freeMem()`, `Image< float >::freeMem()`, `ModelComponent::reset()`, and `SharedPtr< ChannelBase >::reset()`.

Referenced by `setVC()`.

#### 10.251.3.19 `void ShapeEstimator::resetCumMask ()`

reset the cumulative mask

Definition at line 454 of file ShapeEstimator.C.

References `Image< float >::freeMem()`.

#### 10.251.3.20 `void ShapeEstimator::setVC (const SharedPtr< VisualCortex > & vc)`

set the [VisualCortex](#) if it was not given at construction

Definition at line 71 of file ShapeEstimator.C.

References `reset()`.

### 10.251.4 Member Data Documentation

#### 10.251.4.1 `ModelParam<ShapeEstimatorMode> ShapeEstimator::itsMode [protected]`

Determines what information is used as a source for the shape estimation.

possible values:

- "None" - shape estimator is not used;
- "FeatureMap" - the shape is extracted from the winning [Feature Map](#);
- "ConspicuityMap" - the shape is extracted from the winning [Conspicuity Map](#);
- "SaliencyMap" - the shape is extracted from the [Saliency Map](#)

Definition at line 151 of file ShapeEstimator.H.

Referenced by `compute()`, `exportOptions()`, `getIORMask()`, `getObjectMask()`, `getPyramid()`, `getSingleChannel()`, and `getSmoothMask()`.

#### 10.251.4.2 `ModelParam<ShapeEstimatorSmoothMethod> ShapeEstimator::itsSmMethod [protected]`

Determines the Smoothing Method used to extract a map in image coordinates.

possible values:

- "None" - no smoothing, the result is just scaled up in blocks;
- "Gaussian" - smoothing is done by convolving with a large 2D Gaussian kernel;
- "Chamfer" - smoothing is done with Opening and Chamfering (takes about half the time compared to "Gaussian")

Definition at line 158 of file ShapeEstimator.H.

Referenced by `exportOptions()`, and `getSmoothMask()`.

The documentation for this class was generated from the following files:

- [ShapeEstimator.H](#)
- [ShapeEstimator.C](#)

## 10.252 SharedPtr< T > Class Template Reference

A smart pointer with reference counted copy semantics.

```
#include <SharedPtr.H>
```

Collaboration diagram for SharedPtr< T >:

### Public Member Functions

- `SharedPtr (T *p=0)`  
*Construct from a raw pointer. WARNING: do this only ONCE per pointer!*
- `SharedPtr (const SharedPtr &r) throw ()`  
*Copy another SharedPtr. The ref count is incremented by one.*
- `~SharedPtr ()`  
*Destructor decrements the ref count by one.*
- `SharedPtr & operator= (const SharedPtr &r)`  
*Assign from another SharedPtr. The ref count is incremented by one.*
- `template<class TT> SharedPtr (const SharedPtr< TT > &r) throw ()`  
*Copy construct from a SharedPtr of a different type.*
- `template<class TT> SharedPtr & operator= (const SharedPtr< TT > &r)`  
*Assign from a SharedPtr of a different type.*
- `template<class TT> SharedPtr & dynCastFrom (const SharedPtr< TT > &r)`  
*Assign from a SharedPtr of a different type, with dynamic casting.*
- `void reset (T *p=0)`  
*Make the SharedPtr point to a different (optionally null) pointee.*
- `T & operator * () const throw ()`  
*Get a reference to the pointee.*
- `T * operator → () const throw ()`  
*Get the pointee for accessing its members.*
- `T * get () const throw ()`  
*Get the pointee.*

- long `useCount ()` const throw ()  
*Query how many SharedPtr's are sharing the pointee.*
- bool `unique ()` const throw ()  
*Query whether the SharedPtr is the unique owner of its pointee.*
- void `swap (SharedPtr< T > &that)` throw ()  
*Swap the pointees of two SharedPtr's.*

## Friends

- class `SharedPtr`

### 10.252.1 Detailed Description

**template<class T> class SharedPtr< T >**

A smart pointer with reference counted copy semantics.

The object pointed to is deleted when the last SharedPtr pointing to it is destroyed or reset. Take note of the WARNING given in the documentation for the constructor that takes a raw pointer as its argument.

Borrowed and modified from boost.org smart\_ptr.hpp Original copyright notice:

(C) Copyright Greg Colvin and Beman Dawes 1998, 1999. Permission to copy, use, modify, sell and distribute this software is granted provided this copyright notice appears in all copies. This software is provided "as is" without express or implied warranty, and with no claim as to its suitability for any purpose.

Definition at line 55 of file SharedPtr.H.

### 10.252.2 Constructor & Destructor Documentation

**10.252.2.1 template<class T> SharedPtr< T >::SharedPtr (T \* p = 0)**  
[inline, explicit]

Construct from a raw pointer. WARNING: do this only ONCE per pointer!

Bad Things (TM) will happen if this constructor is used multiple times to make multiple SharedPtr's from a single raw pointer; that's because each SharedPtr will start off thinking that it is the unique owner, and so they will each start their own ref count. If multiple SharedPtr's are needed for the same object (after all, that's the whole point of a shared ptr), only the first one should be constructed from the raw pointer, and



subsequent ones should be constructed from the first using the copy constructor or assignment operator.

Definition at line 194 of file SharedPtr.H.

Referenced by SharedPtr< T >::operator=(), and SharedPtr< T >::reset().

**10.252.2.2** `template<class T> SharedPtr< T >::SharedPtr (const SharedPtr< T > & r) throw () [inline]`

Copy another SharedPtr. The ref count is incremented by one.

Definition at line 204 of file SharedPtr.H.

**10.252.2.3** `template<class T> SharedPtr< T >::~~SharedPtr () [inline]`

Destructor decrements the ref count by one.

If we were the last pointer to the pointee, then the pointee and the reference count are both delete'd.

Definition at line 212 of file SharedPtr.H.

**10.252.2.4** `template<class T> template<class TT> SharedPtr< T >::SharedPtr (const SharedPtr< TT > & r) throw () [inline]`

Copy construct from a SharedPtr of a different type.

The other type TT must be related to T by an appropriate inheritance relationship or by a change in const-qualification.

Definition at line 225 of file SharedPtr.H.

## 10.252.3 Member Function Documentation

**10.252.3.1** `template<class T> template<class TT> SharedPtr< T > & SharedPtr< T >::dynCastFrom (const SharedPtr< TT > & r) [inline]`

Assign from a SharedPtr of a different type, with dynamic casting.

This is the SharedPtr equivalent of doing a `dynamic_cast` between two raw pointers. Doing `'SharedPtr ptr; ptr.dynCastFrom(other);'` with `'other'` a SharedPtr will internally do a `dynamic_cast` from `TT*` to `T*` and will assign the result to `ptr`. If the `dynamic_cast` succeeds, `ptr` and `other` will share the pointee as if they were two normal SharedPtr onto it (even though `ptr` and `other` are

SharedPtr of different types). If the dynamic\_cast fails then ptr will point to NULL. Typical usage of this is to recover an expected derived class from a base class: for example,

```
// get our visual cortex from Brain. Brain provides a getVC() function
// which returns a SharedPtr onto the VisualCortex base class no matter
// which derivation of that we are actually running right now:
SharedPtr<VisualCortex> vcx = brain->$getVC();

// do something special if our VisualCortex actually happens to be of
// type VisualCortexEyeMvt which is derived from VisualCortex:
SharedPtr<VisualCortexEyeMvt> vcem;
vcem.dynCastFrom(vcx);

// vcem points to NULL unless the pointee of vcx could be
// dynamic_cast'ed to vcem's type.
if (vcem.get()) {
    // yes, indeed we are running a VisualCortexEyeMvt
    ...
}
```

Note: make sure you always use virtual destructors in your pointees, especially when you use dynCastFrom. Indeed, whichever SharedPtr is the last one to run out of scope will destroy the pointee.

Definition at line 243 of file SharedPtr.H.

References SharedPtr< T >::pn, and SharedPtr< T >::px.

Referenced by SimulationViewerEyeMvt2::attentionShift(), VisualCortex::channelPtr(), RGBConvolveChannel::computePyramid(), SimulationViewerStd::drawMegaCombo(), SimulationViewerStd::drawMotionSixPack(), Brain::evolve(), SimulationViewerEyeMvt::eyeMovement(), SimulationViewerCompress::getTraj(), main(), VisualCortexEyeMvt::start1(), and SimulationViewerCompress::start1().

### 10.252.3.2 template<class T> T\* SharedPtr< T >::get () const throw () [inline]

Get the pointee.

Definition at line 140 of file SharedPtr.H.

Referenced by SimulationViewerEyeMvt2::attentionShift(), VisualBuffer::bufferToRetinal(), VisualCortex::channelPtr(), SimulationViewerStd::channelUpdate(), SingleChannelBeoServer::check(), SimulationViewerStd::drawMegaCombo(), SimulationViewerStd::drawMotionSixPack(), EyeTrackerSaccadeController::evolve(), SimulationViewerEyeMvt::eyeMovement(), SimulationViewerCompress::getTraj(), ModelComponent::hasSubComponent(), main(), operator!=(), operator==(), ModelComponent::removeSubComponent(), VisualBuffer::retinalToBuffer(), VisualCortexEyeMvt::start1(), TaskRelevanceMapKillStatic::start1(), and ModelManager::unregisterComponent().

### 10.252.3.3 `template<class T> T& SharedPtr< T >::operator * () const throw ()` [inline]

Get a reference to the pointee.

Definition at line 134 of file SharedPtr.H.

### 10.252.3.4 `template<class T> T* SharedPtr< T >::operator → () const throw ()` [inline]

Get the pointee for accessing its members.

Definition at line 137 of file SharedPtr.H.

### 10.252.3.5 `template<class T> template<class TT> SharedPtr< T > & SharedPtr< T >::operator= (const SharedPtr< TT > & r)` [inline]

Assign from a SharedPtr of a different type.

The other type TT must be related to T by an appropriate inheritance relationship or by a change in const-qualification.

Definition at line 234 of file SharedPtr.H.

References SharedPtr< T >::SharedPtr().

### 10.252.3.6 `template<class T> SharedPtr< T > & SharedPtr< T >::operator= (const SharedPtr< T > & r)` [inline]

Assign from another SharedPtr. The ref count is incremented by one.

Definition at line 216 of file SharedPtr.H.

References SharedPtr< T >::SharedPtr().

**10.252.3.7** `template<class T> void SharedPtr< T >::reset (T * p = 0)`  
[inline]

Make the SharedPtr point to a different (optionally null) pointee.

Definition at line 263 of file SharedPtr.H.

References `SharedPtr< T >::SharedPtr()`.

Referenced by `SingleChannelBeoServer::check()`, `getSalRegions()`, `main()`, and `VisualCortexEyeMvt::start1()`.

**10.252.3.8** `template<class T> void SharedPtr< T >::swap (SharedPtr< T > & that) throw ()` [inline]

Swap the pointees of two SharedPtr's.

Definition at line 270 of file SharedPtr.H.

**10.252.3.9** `template<class T> bool SharedPtr< T >::unique () const throw ()`  
[inline]

Query whether the SharedPtr is the unique owner of its pointee.

Definition at line 146 of file SharedPtr.H.

**10.252.3.10** `template<class T> long SharedPtr< T >::useCount () const throw ()` [inline]

Query how many SharedPtr's are sharing the pointee.

Definition at line 143 of file SharedPtr.H.

The documentation for this class was generated from the following file:

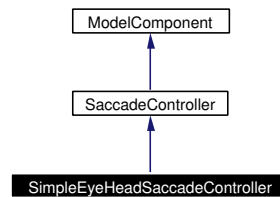
- [SharedPtr.H](#)

## 10.253 SimpleEyeHeadSaccadeController Class Reference

Simple eye/head controller.

```
#include <SaccadeControllers.H>
```

Inheritance diagram for SimpleEyeHeadSaccadeController:



Collaboration diagram for SimpleEyeHeadSaccadeController:

### Public Member Functions

- [SimpleEyeHeadSaccadeController](#) ([ModelManager](#) &mgr)  
*Constructor. See the base classes for parameters.*
- virtual [~SimpleEyeHeadSaccadeController](#) ()  
*Destructor.*
- virtual void [exportOptions](#) (const int whatToExport, const bool recurse=true)  
*Export some of our [ModelParam](#) parameters as command-line-options.*
- virtual void [evolve](#) (const double t)  
*evolve one time step (called, e.g., by [Brain::evolve\(\)](#))*
- [Point2DT](#) [getHead](#) () const  
*Get the current direction of the head.*

### Protected Member Functions

- virtual void [computeWhenNewPercept](#) ()  
*This method is called each time a new percept has arrived.*

- virtual void [computeWhenResetPos](#) ()

*This method is called each time a reset is made.*

- virtual [Point2D](#) [computeWhenNewDecision](#) (const double t)

*This method is called each time a new decision is requested.*

- virtual [Point2D](#) [computeWhenNewDecisionHead](#) (const double t)

*This method is called each time a new head decision is requested.*

### 10.253.1 Detailed Description

Simple eye/head controller.

This fairly trivial controller uses a [ThresholdFrictionSaccadeController](#) to move the eyes, plus a slow [FrictionSaccadeController](#) to move the head. This is a fairly bogus implementation just used here for testing purposes. The eyes basically do all the work, and the head part here just slowly follows the eyes around. The fixations returned are the same as for the [ThresholdFrictionSaccadeController](#). The [getHead\(\)](#) function returns coordinates of where the head is pointed to.

Definition at line 380 of file SaccadeControllers.H.

### 10.253.2 Constructor & Destructor Documentation

#### 10.253.2.1 SimpleEyeHeadSaccadeController::SimpleEyeHeadSaccadeController ([ModelManager](#) & *mgr*)

Constructor. See the base classes for parameters.

Definition at line 655 of file SaccadeControllers.C.

References [ModelComponent::addSubComponent\(\)](#).

#### 10.253.2.2 SimpleEyeHeadSaccadeController::~SimpleEyeHeadSaccadeController () [virtual]

Destructor.

Definition at line 666 of file SaccadeControllers.C.

### 10.253.3 Member Function Documentation

#### 10.253.3.1 [Point2D](#) SimpleEyeHeadSaccadeController::computeWhenNewDecision (const double *t*) [protected, virtual]

This method is called each time a new decision is requested.

Implements [SaccadeController](#).

Definition at line 709 of file SaccadeControllers.C.

References [Point2D::i](#), and [Point2D::j](#).

#### 10.253.3.2 [Point2D](#) SimpleEyeHeadSaccadeController::computeWhenNewDecisionHead (const double *t*) [protected, virtual]

This method is called each time a new head decision is requested.

Reimplemented from [SaccadeController](#).

Definition at line 730 of file SaccadeControllers.C.

References [Point2D::i](#), and [Point2D::j](#).

#### 10.253.3.3 void SimpleEyeHeadSaccadeController::computeWhenNewPercept () [protected, virtual]

This method is called each time a new percept has arrived.

Implements [SaccadeController](#).

Definition at line 692 of file SaccadeControllers.C.

References [SaccadeController::getPreviousPercept\(\)](#).

#### 10.253.3.4 void SimpleEyeHeadSaccadeController::computeWhenResetPos () [protected, virtual]

This method is called each time a reset is made.

Implements [SaccadeController](#).

Definition at line 703 of file SaccadeControllers.C.

**10.253.3.5 void SimpleEyeHeadSaccadeController::evolve (const double t)**  
[virtual]

evolve one time step (called, e.g., by [Brain::evolve\(\)](#))

Implements [SaccadeController](#).

Definition at line 682 of file SaccadeControllers.C.

**10.253.3.6 void SimpleEyeHeadSaccadeController::exportOptions (const int  
whatToExport, const bool recurse = true) [virtual]**

Export some of our [ModelParam](#) parameters as command-line-options.

See the base function in [ModelComponent.H](#) for info.

Reimplemented from [ModelComponent](#).

Definition at line 670 of file SaccadeControllers.C.

References [ModelComponent::exportOptions\(\)](#).

**10.253.3.7 [Point2DT](#) SimpleEyeHeadSaccadeController::getHead () const**

Get the current direction of the head.

The documentation for this class was generated from the following files:

- [SaccadeControllers.H](#)
- [SaccadeControllers.C](#)

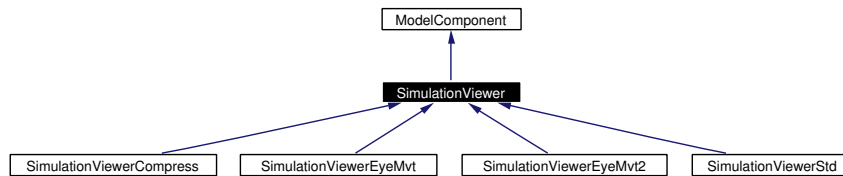


## 10.254 SimulationViewer Class Reference

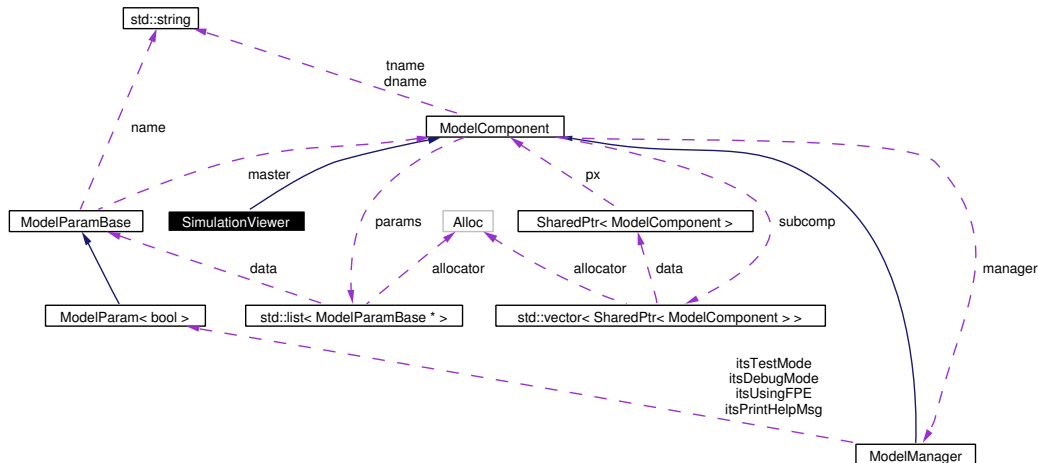
This class provides an abstract interface for a simulation viewer.

```
#include <SimulationViewer.H>
```

Inheritance diagram for SimulationViewer:



Collaboration diagram for SimulationViewer:



## Public Member Functions

### Constructors and destructors

- [SimulationViewer](#) ([ModelManager](#) &mgr, const [std::string](#) &descr-Name="Simulation Viewer", const [std::string](#) &tagName="SimulationViewer")  
*Constructor. See [ModelComponent.H](#).*
- virtual [~SimulationViewer](#) ()

*Destructor.*

- virtual void [setBrain](#) ([Brain](#) \*brain)  
*Set our brain.*

### Events that will trigger new visualizations

- virtual void [newInputReady](#) ()=0  
*Ready to load up a new input frame.*
- virtual void [newInputFrame](#) (const [Image](#)< [PixRGB](#)< [byte](#) > > &input, const double t)=0  
*Got a new input frame at the specified (current) time.*
- virtual void [channelUpdate](#) ([SharedPtr](#)< [VisualCortex](#) > &vcx)=0  
*Update with availability of new channel drawings.*
- virtual void [evolve](#) (const double t)=0  
*Called every simulation time step (time is in seconds).*
- virtual void [attentionShift](#) (const [WTAwinner](#) &win, const [Image](#)< [byte](#) > &foaMask)=0  
*Update with new covert attention position.*
- virtual void [eyeMovement](#) (const [Point2DT](#) &curreye, const [Image](#)< [byte](#) > &foveaMask)=0  
*Update with new eye position.*
- virtual void [headMovement](#) (const [Point2DT](#) &currhead, const [Image](#)< [byte](#) > &headMask)=0  
*Update with new head position.*
- virtual void [eyeSaccade](#) (const bool inSaccade)=0  
*Update eye saccade status.*
- virtual void [headSaccade](#) (const bool inSaccade)=0  
*Update head saccade status.*
- virtual void [eyeBlink](#) (const bool inBlink)=0  
*Update eye blink status.*

### Output of our current internal drawings

- virtual [Image](#)< [PixRGB](#)< [byte](#) > > [getTraj](#) (const double t)=0  
*Get the attention/eye/head trajectory image.*

- virtual void [saveResults](#) (const [SharedPtr](#)< [OutputFrameSeries](#) > &ofs)=0  
*Save our various results.*

### 10.254.1 Detailed Description

This class provides an abstract interface for a simulation viewer.

Its typical usage is with a model that generates shifts of attention, eye and head movements. Derivatives from this class will handle visualizing those events. See [SimulationViewerStd.H](#) for an example of such derivative. The base class described here just specifies the interface.

Definition at line 58 of file [SimulationViewer.H](#).

### 10.254.2 Constructor & Destructor Documentation

#### 10.254.2.1 [SimulationViewer::SimulationViewer](#) ([ModelManager](#) & mgr, const [std::string](#) & descrName = "Simulation Viewer", const [std::string](#) & tagName = "SimulationViewer")

Constructor. See [ModelComponent.H](#).

##### Parameters:

- mgr* our [ModelManager](#) (see [ModelManager.H](#))
- descrName* descriptive name for human useag
- tagName* name for [ParamMap](#) usage

Definition at line 46 of file [SimulationViewer.C](#).

#### 10.254.2.2 [SimulationViewer::~~SimulationViewer](#) () [virtual]

Destructor.

Definition at line 53 of file [SimulationViewer.C](#).

### 10.254.3 Member Function Documentation

#### 10.254.3.1 virtual void [SimulationViewer::attentionShift](#) (const [WTAwinner](#) & win, const [Image](#)< [byte](#) > & foaMask) [pure virtual]

Update with new covert attention position.

**Parameters:**

**win** the coordinates and other info about this new covert shift of attention

**foaMask** an image that is all-zero except that it is 255 inside a region that corresponds to the shape and size of the current focus of attention. If such image is not available, an uninitialized image should be passed here, otherwise it should have same dims as the input frames.

Implemented in [SimulationViewerCompress](#), [SimulationViewerEyeMvt](#), [SimulationViewerEyeMvt2](#), and [SimulationViewerStd](#).

#### 10.254.3.2 **virtual void SimulationViewer::channelUpdate** ([SharedPtr< VisualCortex >](#) &vcx) [pure virtual]

Update with availability of new channel drawings.

**Parameters:**

**vcx** The [VisualCortex](#) pointer allows us to access the channels if we need to.

Implemented in [SimulationViewerCompress](#), [SimulationViewerEyeMvt](#), [SimulationViewerEyeMvt2](#), and [SimulationViewerStd](#).

#### 10.254.3.3 **virtual void SimulationViewer::evolve** (const double t) [pure virtual]

Called every simulation time step (time is in seconds).

Implemented in [SimulationViewerCompress](#), [SimulationViewerEyeMvt](#), [SimulationViewerEyeMvt2](#), and [SimulationViewerStd](#).

#### 10.254.3.4 **virtual void SimulationViewer::eyeBlink** (const bool inBlink) [pure virtual]

Update eye blink status.

inBlink should be true when a blink starts and false when it ends

Implemented in [SimulationViewerCompress](#), [SimulationViewerEyeMvt](#), [SimulationViewerEyeMvt2](#), and [SimulationViewerStd](#).

#### 10.254.3.5 **virtual void SimulationViewer::eyeMovement** (const [Point2DT](#) &curreye, const [Image< byte >](#) &foveaMask) [pure virtual]

Update with new eye position.

**Parameters:**

- curreye*** the coordinates and time of this new overt shift of attention (=eye movement)
- foveaMask*** an image that is all-zero except that it is 255 inside a region that corresponds to the shape and size of the current fovea. If such image is not available, an uninitialized image should be passed here, otherwise it should have same dims as the input frames.

Implemented in [SimulationViewerCompress](#), [SimulationViewerEyeMvt](#), [SimulationViewerEyeMvt2](#), and [SimulationViewerStd](#).

### 10.254.3.6 virtual void SimulationViewer::eyeSaccade (const bool *inSaccade*) [pure virtual]

Update eye saccade status.

*inSaccade* should be true when a saccade starts and false when it ends

Implemented in [SimulationViewerCompress](#), [SimulationViewerEyeMvt](#), [SimulationViewerEyeMvt2](#), and [SimulationViewerStd](#).

### 10.254.3.7 virtual Image< PixRGB<byte> > SimulationViewer::getTraj (const double *t*) [pure virtual]

Get the attention/eye/head trajectory image.

This is called internally by [saveResults\(\)](#), which should normally be used. It is provided here for those models which do not use [FrameSeries](#) and hence cannot call [saveResults\(\)](#).

Implemented in [SimulationViewerCompress](#), [SimulationViewerEyeMvt](#), [SimulationViewerEyeMvt2](#), and [SimulationViewerStd](#).

### 10.254.3.8 virtual void SimulationViewer::headMovement (const Point2DT & *currhead*, const Image< byte > & *headMask*) [pure virtual]

Update with new head position.

**Parameters:**

- currhead*** the coordinates and time of the new head position
- headMask*** an image that is all-zero except that it is 255 inside a region that corresponds to the shape and size of the current head marker. If such image is not available, an uninitialized image should be passed here, otherwise it should have same dims as the input frames.

Implemented in [SimulationViewerCompress](#), [SimulationViewerEyeMvt](#), [SimulationViewerEyeMvt2](#), and [SimulationViewerStd](#).

#### 10.254.3.9 **virtual void SimulationViewer::headSaccade (const bool *inSaccade*)** [pure virtual]

Update head saccade status.

*inSaccade* should be true when a saccade starts and false when it ends

Implemented in [SimulationViewerCompress](#), [SimulationViewerEyeMvt](#), [SimulationViewerEyeMvt2](#), and [SimulationViewerStd](#).

#### 10.254.3.10 **virtual void SimulationViewer::newInputFrame (const [Image](#)<[PixRGB](#)<[byte](#)>> & *input*, const double *t*)** [pure virtual]

Got a new input frame at the specified (current) time.

This is called by [Brain](#) after the frame is loaded into the [Brain](#), [VisualCortex](#) and others, and after all the low-level convolutions have been made.

##### Parameters:

*input* the new input image

*t* the current time, in seconds

Implemented in [SimulationViewerCompress](#), [SimulationViewerEyeMvt](#), [SimulationViewerEyeMvt2](#), and [SimulationViewerStd](#).

#### 10.254.3.11 **virtual void SimulationViewer::newInputReady ()** [pure virtual]

Ready to load up a new input frame.

This is called by [Brain](#) just before a new input frame is being loaded into [Brain](#). Thus, this gives a last opportunity for visualizations on the current input, just before it gets replaced by the new one.

Implemented in [SimulationViewerCompress](#), [SimulationViewerEyeMvt](#), [SimulationViewerEyeMvt2](#), and [SimulationViewerStd](#).

#### 10.254.3.12 **virtual void SimulationViewer::saveResults (const [SharedPtr](#)<[OutputFrameSeries](#)> & *ofs*)** [pure virtual]

Save our various results.

**Parameters:**

*ofs* this will figure out file names and formats and save our results

Implemented in [SimulationViewerCompress](#), [SimulationViewerEyeMvt](#), [SimulationViewerEyeMvt2](#), and [SimulationViewerStd](#).

**10.254.3.13 void SimulationViewer::setBrain ([Brain](#) \* *brain*) [virtual]**

Set our brain.

Some viewings may require a [Brain](#). Since in the base class we don't have a [Brain](#) data member, the base class implementation of this function is a no-op. Derived class should overload this accordingly if they need a [Brain](#).

Reimplemented in [SimulationViewerCompress](#), [SimulationViewerEyeMvt](#), [SimulationViewerEyeMvt2](#), and [SimulationViewerStd](#).

Definition at line 57 of file [SimulationViewer.C](#).

The documentation for this class was generated from the following files:

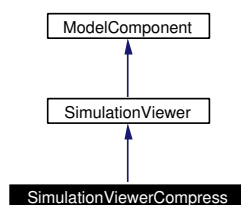
- [SimulationViewer.H](#)
- [SimulationViewer.C](#)

## 10.255 SimulationViewerCompress Class Reference

Do multi-foveated saliency-based image compression.

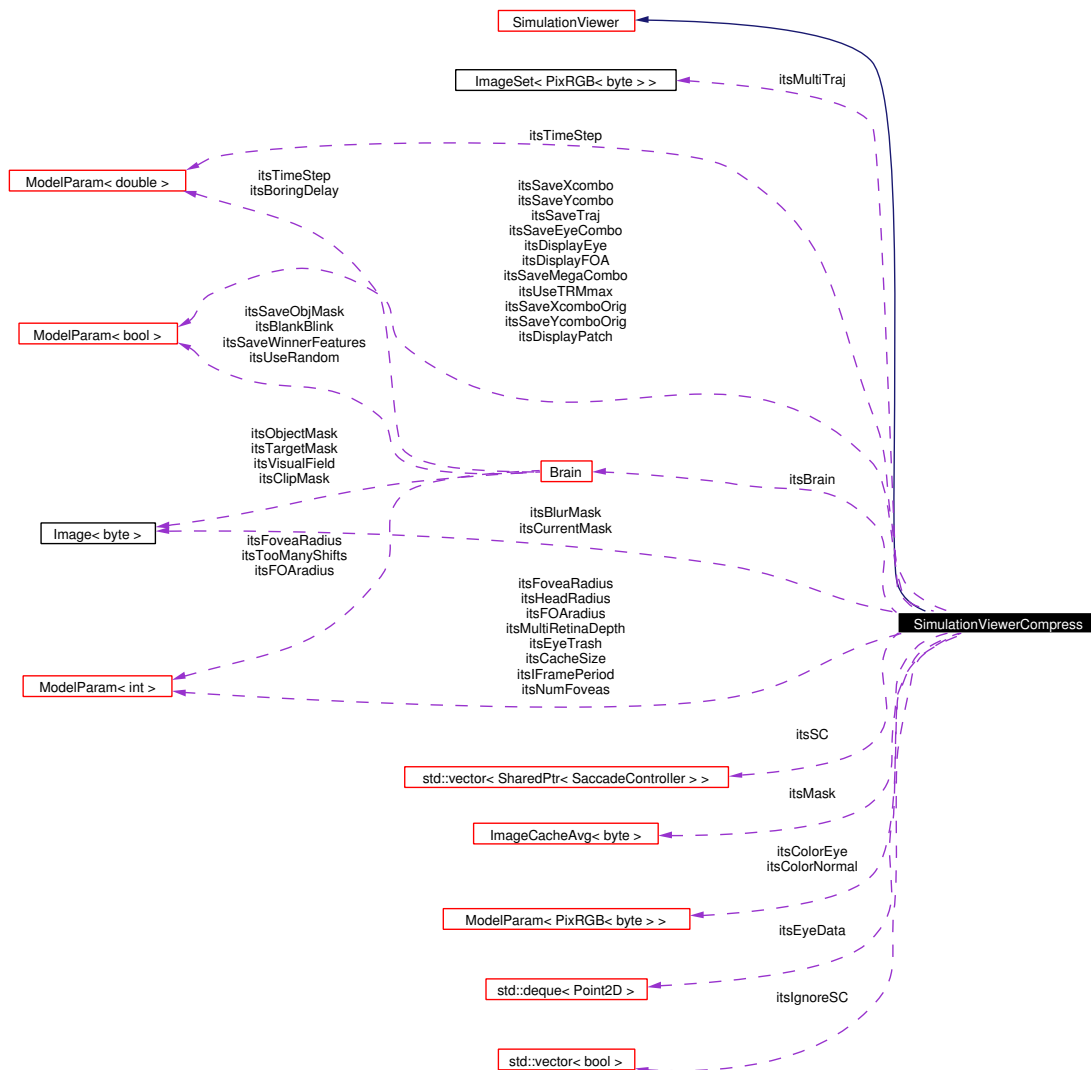
```
#include <SimulationViewerCompress.H>
```

Inheritance diagram for SimulationViewerCompress:



Collaboration diagram for SimulationViewerCompress:





## Public Member Functions

### Constructors and destructors

- [SimulationViewerCompress](#) ([ModelManager](#) &mgr, const [std::string](#) &descr-Name="Multi-Foveated Compression Simulation Viewer", const [std::string](#) &tagName="SimulationViewerCompress")

Constructor. See [ModelComponent.H](#).

- virtual `~SimulationViewerCompress ()`  
*Destructor.*
- virtual void `exportOptions (const int whatToExport, const bool recurse=true)`  
*Export some of our [ModelParam](#) parameters as command-line-options.*
- virtual void `setBrain (Brain *brain)`  
*Set our brain.*

### Events that will trigger new visualizations

- virtual void `newInputReady ()`  
*Ready to switch to next input frame.*
- virtual void `newInputFrame (const Image< PixRGB< byte > > &input, const double t)`  
*Got a new input frame at the specified (current) time.*
- virtual void `channelUpdate (SharedPtr< VisualCortex > &vcx)`  
*Update with availability of new channel drawings.*
- virtual void `evolve (const double t)`  
*Called every simulation time step (time is in seconds).*
- virtual void `attentionShift (const WTAwinner &win, const Image< byte > &foaMask)`  
*Update with new covert attention position.*
- virtual void `eyeMovement (const Point2DT &curreye, const Image< byte > &foveaMask)`  
*Update with new eye position.*
- virtual void `headMovement (const Point2DT &currhead, const Image< byte > &headMask)`  
*Update with new head position.*
- virtual void `eyeSaccade (const bool inSaccade)`  
*Update eye saccade status.*
- virtual void `headSaccade (const bool inSaccade)`  
*Update head saccade status.*
- virtual void `eyeBlink (const bool inBlink)`  
*Update eye blink status.*

### Output of our current internal drawings

- virtual [Image](#)< [PixRGB](#)< [byte](#) > > [getTraj](#) (const double t)  
*Get the attention/eye/head trajectory image.*
- virtual void [saveResults](#) (const [SharedPtr](#)< [OutputFrameSeries](#) > &ofs)  
*Save our various results.*

### Protected Member Functions

- virtual void [start1](#) ()  
*Get started and disable any SC the [Brain](#) may want to use.*
- virtual void [stop1](#) ()  
*get stopped*
- virtual bool [setModelParamString](#) (const [std::string](#) &name, const [std::string](#) &value, const bool recurse=false, const bool warn=true)  
*Intercept people changing our number of foveas.*

### Protected Attributes

- [ModelParam](#)< int > [itsNumFoveas](#)  
*number of foveas*
- [ModelParam](#)< bool > [itsSaveTraj](#)  
*save trajectory?*
- [ModelParam](#)< bool > [itsSaveXcombo](#)  
*save trajectory+mask?*
- [ModelParam](#)< bool > [itsSaveYcombo](#)  
*save trajectory+mask?*
- [ModelParam](#)< bool > [itsSaveXcomboOrig](#)  
*save original+trajectory?*
- [ModelParam](#)< bool > [itsSaveYcomboOrig](#)  
*save original+trajectory?*

- [ModelParam](#)< bool > [itsSaveMegaCombo](#)  
*save mega combo?*
- [ModelParam](#)< bool > [itsSaveEyeCombo](#)  
*save eye combo?*
- [ModelParam](#)< bool > [itsDisplayPatch](#)  
*draw patches*
- [ModelParam](#)< bool > [itsDisplayFOA](#)  
*draw objetc outlines*
- [ModelParam](#)< bool > [itsDisplayEye](#)  
*draw human eye movements*
- [ModelParam](#)< [PixRGB](#)< byte > > [itsColorNormal](#)  
*patch color*
- [ModelParam](#)< [PixRGB](#)< byte > > [itsColorEye](#)  
*patch color for human eye mvts*
- [ModelParam](#)< int > [itsFOAradius](#)  
*radius of FOA*
- [ModelParam](#)< int > [itsFoveaRadius](#)  
*radius of Fovea*
- [ModelParam](#)< int > [itsHeadRadius](#)  
*head radius*
- [ModelParam](#)< int > [itsMultiRetinaDepth](#)  
*depth of blur pyramid*
- [ModelParam](#)< double > [itsTimeStep](#)  
*simulation time step*
- [ModelParam](#)< int > [itsCacheSize](#)  
*size of our mask cache*
- [ModelParam](#)< bool > [itsUseTRMmax](#)  
*use TRM to take max in cache*

- [ModelParam< std::string > itsFoveaSCtype](#)  
*type of SC for foveas*
- [ModelParam< std::string > itsOutFname](#)  
*Name of output file.*
- [ModelParam< int > itsEyeTrash](#)  
*how many initial eye samples trash*
- [ModelParam< int > itsIframePeriod](#)  
*This parameter is the period (in frames) for foveation mask to change.*

### 10.255.1 Detailed Description

Do multi-foveated saliency-based image compression.

This viewer will consider the top N salient locations and apply blurring to the image that increases in strength (sigma) as we get farther away from any of the top N most salient locations. The resulting trajectory image hence will be crisp at the top N locations and increasingly blurred away from those. Compressing the output of this viewer, e.g., using MPEG, yields smaller file size while (hopefully) preserving high quality at the important image locations. This viewer relies on a collection of N Saccade-Controllers to track the N hotspots; you can pick which type of controller to use and configure their parameters as usual, via the command line. Note that in programs like ezvision that use an [StdBrain](#), the [StdBrain](#) also contains a [SaccadeController](#); but here we will explicitly drop it, as we won't be using Brain's controller, since we will decide on the N hotspots here based on the raw saliency map. We will also drop Brain's [ShapeEstimator](#) and force Brain's IOR type to none, so that we have a clean saliency map here to work with.

Definition at line 64 of file SimulationViewerCompress.H.

### 10.255.2 Constructor & Destructor Documentation

**10.255.2.1 SimulationViewerCompress::SimulationViewerCompress**  
([ModelManager](#) & mgr, const [std::string](#) & descrName =  
"Multi-Foveated Compression Simulation Viewer", const [std::string](#)  
& tagName = "SimulationViewerCompress")

Constructor. See [ModelComponent.H](#).

Definition at line 65 of file SimulationViewerCompress.C.

References [OPTEXP\\_NONE](#), and [ModelManager::setOptionValString\(\)](#).

### 10.255.2.2 **SimulationViewerCompress::~~SimulationViewerCompress ()** [virtual]

Destructor.

Definition at line 108 of file SimulationViewerCompress.C.

## 10.255.3 Member Function Documentation

### 10.255.3.1 **void SimulationViewerCompress::attentionShift (const [WTWinner](#) & win, const [Image](#)< [byte](#) > & fofMask)** [virtual]

Update with new covert attention position.

In SimulationViewerCompress, this is a no-op.

Implements [SimulationViewer](#).

Definition at line 285 of file SimulationViewerCompress.C.

### 10.255.3.2 **void SimulationViewerCompress::channelUpdate ([SharedPtr](#)< [VisualCortex](#) > & vcx)** [virtual]

Update with availability of new channel drawings.

In SimulationViewerCompress, this is a no-op.

Implements [SimulationViewer](#).

Definition at line 261 of file SimulationViewerCompress.C.

### 10.255.3.3 **void SimulationViewerCompress::evolve (const double t)** [virtual]

Called every simulation time step (time is in seconds).

In SimulationViewerCompress, we evolve the positions of our internal foveas.

Implements [SimulationViewer](#).

Definition at line 265 of file SimulationViewerCompress.C.

### 10.255.3.4 **void SimulationViewerCompress::exportOptions (const int whatToExport, const bool recurse = true)** [virtual]

Export some of our [ModelParam](#) parameters as command-line-options.

See the base function in [ModelComponent.H](#) for info. Actually, here we just keep a copy of whatToExport and we will call [exportOptions\(\)](#) on our [SaccadeController](#) once we know its type and instantiate it.

Reimplemented from [ModelComponent](#).

Definition at line 166 of file SimulationViewerCompress.C.

References [ModelComponent::exportOptions\(\)](#), [itsCacheSize](#), [itsDisplayEye](#), [itsDisplayFOA](#), [itsDisplayPatch](#), [itsFoveaSCtype](#), [itsHeadRadius](#), [itsIframePeriod](#), [itsMultiRetinaDepth](#), [itsNumFoveas](#), [itsOutFname](#), [itsSaveEyeCombo](#), [itsSaveMegaCombo](#), [itsSaveTraj](#), [itsSaveXcombo](#), [itsSaveXcomboOrig](#), [itsSaveYcombo](#), [itsSaveYcomboOrig](#), [itsUseTRMmax](#), [OPTEXP\\_CORE](#), and [ModelManager::requestOption\(\)](#).

#### 10.255.3.5 void SimulationViewerCompress::eyeBlink (const bool *inBlink*) [virtual]

Update eye blink status.

In SimulationViewerCompress, this is a no-op.

Implements [SimulationViewer](#).

Definition at line 313 of file SimulationViewerCompress.C.

#### 10.255.3.6 void SimulationViewerCompress::eyeMovement (const [Point2DT](#) & *curreye*, const [Image](#)< [byte](#) > & *foveaMask*) [virtual]

Update with new eye position.

In SimulationViewerCompress, this is a no-op.

Implements [SimulationViewer](#).

Definition at line 290 of file SimulationViewerCompress.C.

References [Point2D::i](#), [Point2D::j](#), and [Point2DT::p](#).

#### 10.255.3.7 void SimulationViewerCompress::eyeSaccade (const bool *inSaccade*) [virtual]

Update eye saccade status.

In SimulationViewerCompress, this is a no-op.

Implements [SimulationViewer](#).

Definition at line 305 of file SimulationViewerCompress.C.

### 10.255.3.8 **Image< PixRGB< byte > > SimulationViewerCompress::getTraj** (const double t) [virtual]

Get the attention/eye/head trajectory image.

Implements [SimulationViewer](#).

Definition at line 317 of file SimulationViewerCompress.C.

References [binaryReverse\(\)](#), [Rectangle::bottom\(\)](#), [byte](#), [Point2D::clampToDims\(\)](#), [Image< T >::clear\(\)](#), [concatX\(\)](#), [concatY\(\)](#), [contour2D8\(\)](#), [crop\(\)](#), [decXY\(\)](#), [downSize\(\)](#), [Image< T >::drawDisk\(\)](#), [Image< T >::drawGrid\(\)](#), [Image< T >::drawLine\(\)](#), [Image< T >::drawPatch\(\)](#), [SharedPtr< T >::dynCastFrom\(\)](#), [Rectangle::fitToDims\(\)](#), [SharedPtr< T >::get\(\)](#), [SharedPtr< Retina >::get\(\)](#), [SharedPtr< TaskRelevanceMap >::get\(\)](#), [SharedPtr< AttentionGuidanceMap >::get\(\)](#), [Brain::getAGM\(\)](#), [Image< T >::getDims\(\)](#), [Image< T >::getHeight\(\)](#), [Image< T >::getMinMaxAvg\(\)](#), [ModelComponent::getModelParamVal\(\)](#), [Brain::getRawInput\(\)](#), [Brain::getRET\(\)](#), [Brain::getSM\(\)](#), [Brain::getTRM\(\)](#), [Image< T >::getVal\(\)](#), [ModelParam< PixRGB< byte > >::getVal\(\)](#), [ModelParam< int >::getVal\(\)](#), [ModelParam< bool >::getVal\(\)](#), [Brain::getVC\(\)](#), [Image< T >::getWidth\(\)](#), [Dims::h\(\)](#), [Point2D::i](#), [Image< byte >::initialized\(\)](#), [itsCacheSize](#), [itsColorEye](#), [itsColorNormal](#), [itsDisplayEye](#), [itsDisplayFOA](#), [itsDisplayPatch](#), [itsIFramePeriod](#), [itsSaveEyeCombo](#), [itsSaveMegaCombo](#), [itsSaveXcombo](#), [itsSaveXcomboOrig](#), [itsSaveYcombo](#), [itsSaveYcomboOrig](#), [itsUseTRMmax](#), [Point2D::j](#), [Rectangle::left\(\)](#), [lowPass3\(\)](#), [Image< T >::lowThresh\(\)](#), [LevelSpec::mapLevel\(\)](#), [ImageCacheAvg< byte >::mean\(\)](#), [NO\\_INIT](#), [NRND](#), [Image< T >::paste\(\)](#), [ImageCache< byte >::push\\_back\(\)](#), [randomUpTo\(\)](#), [rescale\(\)](#), [Rectangle::right\(\)](#), [takeMin\(\)](#), [thresholdedMix\(\)](#), [Rectangle::top\(\)](#), [toRGB\(\)](#), [uint](#), [Dims::w\(\)](#), and [weightedBlur\(\)](#).

Referenced by [saveResults\(\)](#).

### 10.255.3.9 **void SimulationViewerCompress::headMovement** (const [Point2DT](#) & curthead, const [Image< byte >](#) & headMask) [virtual]

Update with new head position.

In [SimulationViewerCompress](#), this is a no-op.

Implements [SimulationViewer](#).

Definition at line 300 of file SimulationViewerCompress.C.

### 10.255.3.10 **void SimulationViewerCompress::headSaccade** (const bool inSaccade) [virtual]

Update head saccade status.

In [SimulationViewerCompress](#), this is a no-op.



Implements [SimulationViewer](#).

Definition at line 309 of file SimulationViewerCompress.C.

**10.255.3.11 void SimulationViewerCompress::newInputFrame (const [Image](#)< [PixRGB](#)< [byte](#) > > & input, const double t) [virtual]**

Got a new input frame at the specified (current) time.

In SimulationViewerCompress, this is a no-op.

Implements [SimulationViewer](#).

Definition at line 248 of file SimulationViewerCompress.C.

References [buildPyrGaussian\(\)](#), [ModelParam< int >::getVal\(\)](#), and [itsMultiRetinaDepth](#).

**10.255.3.12 void SimulationViewerCompress::newInputReady () [virtual]**

Ready to switch to next input frame.

Implements [SimulationViewer](#).

Definition at line 244 of file SimulationViewerCompress.C.

**10.255.3.13 void SimulationViewerCompress::saveResults (const [SharedPtr](#)< [OutputFrameSeries](#) > & ofs) [virtual]**

Save our various results.

Implements [SimulationViewer](#).

Definition at line 848 of file SimulationViewerCompress.C.

References [getTraj\(\)](#), [ModelParam< bool >::getVal\(\)](#), [itsSaveEyeCombo](#), [itsSaveMegaCombo](#), [itsSaveTraj](#), [itsSaveXcombo](#), [itsSaveXcomboOrig](#), [itsSaveYcombo](#), and [itsSaveYcomboOrig](#).

**10.255.3.14 void SimulationViewerCompress::setBrain ([Brain](#) \* brain) [virtual]**

Set our brain.

Reimplemented from [SimulationViewer](#).

Definition at line 206 of file SimulationViewerCompress.C.

**10.255.3.15** `bool SimulationViewerCompress::setModelParamString (const std::string & name, const std::string & value, const bool recurse = false, const bool warn = true) [protected, virtual]`

Intercept people changing our number of foveas.

Reimplemented from [ModelComponent](#).

Definition at line 112 of file SimulationViewerCompress.C.

References [ModelParamBase::getName\(\)](#), [ModelParam< int >::getValString\(\)](#), [itsFoveaSCtype](#), [itsNumFoveas](#), and [ModelComponent::setModelParamString\(\)](#).

**10.255.3.16** `void SimulationViewerCompress::start1 () [protected, virtual]`

Get started and disable any SC the [Brain](#) may want to use.

Reimplemented from [ModelComponent](#).

Definition at line 210 of file SimulationViewerCompress.C.

References [SharedPtr< T >::dynCastFrom\(\)](#), [ModelParam< int >::getVal\(\)](#), [itsCacheSize](#), [itsOutFname](#), [ModelComponent::numSubComp\(\)](#), [ImageCache< byte >::setMaxSize\(\)](#), [ModelComponent::subComponent\(\)](#), and [uint](#).

**10.255.3.17** `void SimulationViewerCompress::stop1 () [protected, virtual]`

get stopped

Reimplemented from [ModelComponent](#).

Definition at line 238 of file SimulationViewerCompress.C.

## 10.255.4 Member Data Documentation

**10.255.4.1** `ModelParam<int> SimulationViewerCompress::itsCacheSize [protected]`

size of our mask cache

Definition at line 174 of file SimulationViewerCompress.H.

Referenced by [exportOptions\(\)](#), [getTraj\(\)](#), and [start1\(\)](#).

**10.255.4.2** [ModelParam< PixRGB<byte> > SimulationViewerCompress::its-ColorEye](#) [protected]

patch color for human eye mvts

Definition at line 168 of file SimulationViewerCompress.H.

Referenced by getTraj().

**10.255.4.3** [ModelParam< PixRGB<byte> > SimulationViewerCompress::its-ColorNormal](#) [protected]

patch color

Definition at line 167 of file SimulationViewerCompress.H.

Referenced by getTraj().

**10.255.4.4** [ModelParam<bool> SimulationViewerCompress::itsDisplayEye](#) [protected]

draw human eye movements

Definition at line 166 of file SimulationViewerCompress.H.

Referenced by exportOptions(), and getTraj().

**10.255.4.5** [ModelParam<bool> SimulationViewerCompress::itsDisplayFOA](#) [protected]

draw objetc outlines

Definition at line 165 of file SimulationViewerCompress.H.

Referenced by exportOptions(), and getTraj().

**10.255.4.6** [ModelParam<bool> SimulationViewerCompress::itsDisplayPatch](#) [protected]

draw patches

Definition at line 164 of file SimulationViewerCompress.H.

Referenced by exportOptions(), and getTraj().

**10.255.4.7   [ModelParam<int>](#)   [SimulationViewerCompress::itsEyeTrash](#)  
[protected]**

how many initial eye samples trash

Definition at line 178 of file SimulationViewerCompress.H.

**10.255.4.8   [ModelParam<int>](#)   [SimulationViewerCompress::itsFOAradius](#)  
[protected]**

radius of FOA

Definition at line 169 of file SimulationViewerCompress.H.

**10.255.4.9   [ModelParam<int>](#)   [SimulationViewerCompress::itsFoveaRadius](#)  
[protected]**

radius of Fovea

Definition at line 170 of file SimulationViewerCompress.H.

**10.255.4.10   [ModelParam<std::string>](#)   [SimulationViewerCompress::itsFovea-  
SCtype](#)   [protected]**

type of SC for foveas

Definition at line 176 of file SimulationViewerCompress.H.

Referenced by exportOptions(), and setModelParamString().

**10.255.4.11   [ModelParam<int>](#)   [SimulationViewerCompress::itsHeadRadius](#)  
[protected]**

head radius

Definition at line 171 of file SimulationViewerCompress.H.

Referenced by exportOptions().

**10.255.4.12   [ModelParam<int>](#)   [SimulationViewerCompress::itsIFramePeriod](#)  
[protected]**

This parameter is the period (in frames) for foveation mask to change.

The idea here is to see whether keeping the foveation mask stable for predicted frames and changing it only for intra-coded frames may improve

the compression ratio when using MPEG-1 compression. Have a look at [http://www.disctronics.co.uk/technology/video/video\\_-mpeg.htm](http://www.disctronics.co.uk/technology/video/video_-mpeg.htm) if you are not familiar with those various types of frames. In particular (quoted from that web site):

"I-frames (Intra coded frames) use DCT encoding only to compress a single frame without reference to any other frame in the sequence. [...] P-frames (Predicted frames) are coded as differences from the last I or P frame. The new P-frame is first predicted by taking the last I or P frame and 'predicting' the values of each new pixel. P-frames use Motion Prediction and DCT encoding. As a result P-frames will give a compression ratio better than I-frames but depending on the amount of motion present. The differences between the predicted and actual values are encoded. [...] B-frames (Bidirectional frames) are coded as differences from the last or next I or P frame. B-frames use prediction as for P-frames but for each block either the previous I or P frame is used or the next I or P frame. [...]"

So, in our case, changing the foveation mask on a P or B frame may yield lots of prediction errors, that can be reduced if we force the mask to be only allowed to change on I frames. Here we assume that the first frame is an I-frame.

Definition at line 208 of file SimulationViewerCompress.H.

Referenced by exportOptions(), and getTraj().

#### 10.255.4.13 **ModelParam<int> SimulationViewerCompress::itsMultiRetinaDepth** [protected]

depth of blur pyramid

Definition at line 172 of file SimulationViewerCompress.H.

Referenced by exportOptions(), and newInputFrame().

#### 10.255.4.14 **ModelParam<int> SimulationViewerCompress::itsNumFoveas** [protected]

number of foveas

Definition at line 156 of file SimulationViewerCompress.H.

Referenced by exportOptions(), and setModelParamString().

#### 10.255.4.15 **ModelParam<std::string> SimulationViewerCompress::itsOutFname** [protected]

Name of output file.

Definition at line 177 of file SimulationViewerCompress.H.

Referenced by exportOptions(), and start1().

**10.255.4.16** **ModelParam<bool>** **SimulationViewerCompress::itsSaveEye-Combo** [protected]

save eye combo?

Definition at line 163 of file SimulationViewerCompress.H.

Referenced by exportOptions(), getTraj(), and saveResults().

**10.255.4.17** **ModelParam<bool>** **SimulationViewerCompress::itsSaveMega-Combo** [protected]

save mega combo?

Definition at line 162 of file SimulationViewerCompress.H.

Referenced by exportOptions(), getTraj(), and saveResults().

**10.255.4.18** **ModelParam<bool>** **SimulationViewerCompress::itsSaveTraj** [protected]

save trajectory?

Definition at line 157 of file SimulationViewerCompress.H.

Referenced by exportOptions(), and saveResults().

**10.255.4.19** **ModelParam<bool>** **SimulationViewerCompress::itsSaveXcombo** [protected]

save trajectory+mask?

Definition at line 158 of file SimulationViewerCompress.H.

Referenced by exportOptions(), getTraj(), and saveResults().

**10.255.4.20** **ModelParam<bool>** **SimulationViewerCompress::itsSaveXcombo-Orig** [protected]

save original+trajectory?

Definition at line 160 of file SimulationViewerCompress.H.

Referenced by exportOptions(), getTraj(), and saveResults().

**10.255.4.21** [ModelParam<bool> SimulationViewerCompress::itsSaveYcombo](#)  
[protected]

save trajectory+mask?

Definition at line 159 of file SimulationViewerCompress.H.

Referenced by exportOptions(), getTraj(), and saveResults().

**10.255.4.22** [ModelParam<bool> SimulationViewerCompress::itsSaveYcombo-Orig](#) [protected]

save original+trajectory?

Definition at line 161 of file SimulationViewerCompress.H.

Referenced by exportOptions(), getTraj(), and saveResults().

**10.255.4.23** [ModelParam<double> SimulationViewerCompress::itsTimeStep](#)  
[protected]

simulation time step

Definition at line 173 of file SimulationViewerCompress.H.

**10.255.4.24** [ModelParam<bool> SimulationViewerCompress::itsUseTRMmax](#)  
[protected]

use TRM to take max in cache

Definition at line 175 of file SimulationViewerCompress.H.

Referenced by exportOptions(), and getTraj().

The documentation for this class was generated from the following files:

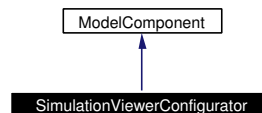
- [SimulationViewerCompress.H](#)
- [SimulationViewerCompress.C](#)

## 10.256 SimulationViewerConfigurator Class Reference

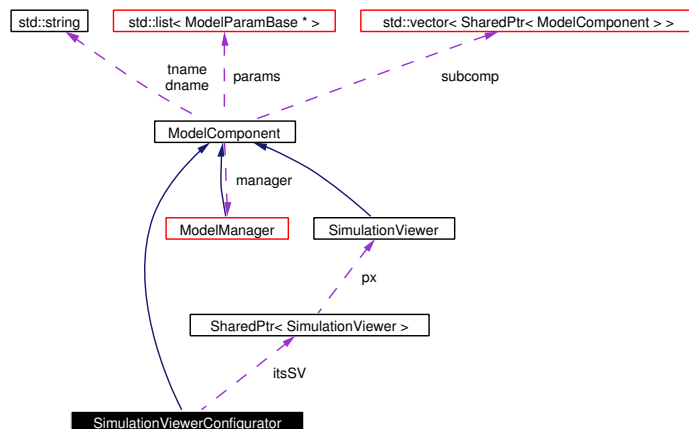
[SimulationViewer](#) configurator.

```
#include <SimulationViewer.H>
```

Inheritance diagram for SimulationViewerConfigurator:



Collaboration diagram for SimulationViewerConfigurator:



### Public Member Functions

- [SimulationViewerConfigurator](#) ([ModelManager](#) &mgr, const [std::string](#) &descrName="Simulation Viewer Configurator", const [std::string](#) &tagName="SimulationViewerConfigurator")

*Constructor.*

- virtual [~SimulationViewerConfigurator](#) ()

*destructor*

- virtual void [exportOptions](#) (const int whatToExport, const bool recurse=true)



Export some of our [ModelParam](#) parameters as command-line-options.

- [SharedPtr](#)< [SimulationViewer](#) > [getSV](#) () const

Get the chosen [SimulationViewer](#).

## Protected Member Functions

- virtual bool [setModelParamString](#) (const [std::string](#) &name, const [std::string](#) &value, const bool recurse=false, const bool warn=true)

Intercept people changing our [ModelParam](#).

## Protected Attributes

- [ModelParam](#)< [std::string](#) > [itsSVtype](#)

type of viewer

### 10.256.1 Detailed Description

[SimulationViewer](#) configurator.

This will export the `-sv-type=XX` command-line option and will instantiate a [SimulationViewer](#) of the desired type as the option gets assigned a value. As this happens, new options may become available in the command-line. To see them, use `-help AFTER` you have chosen the type to use. The current [SimulationViewer](#) may be retrieved using [getSV\(\)](#).

Definition at line 186 of file [SimulationViewer.H](#).

### 10.256.2 Constructor & Destructor Documentation

- 10.256.2.1 [SimulationViewerConfigurator::SimulationViewerConfigurator](#)**  
**([ModelManager](#) & mgr, const [std::string](#) & descrName =**  
**"Simulation Viewer Configurator", const [std::string](#) & tagName =**  
**"SimulationViewerConfigurator")**

Constructor.

Definition at line 64 of file [SimulationViewer.C](#).

References [OPTEXP\\_NONE](#).

### 10.256.2.2 `SimulationViewerConfigurator::~~SimulationViewerConfigurator ()` [virtual]

destructor

Definition at line 73 of file `SimulationViewer.C`.

## 10.256.3 Member Function Documentation

### 10.256.3.1 `void SimulationViewerConfigurator::exportOptions (const int whatToExport, const bool recurse = true) [virtual]`

Export some of our [ModelParam](#) parameters as command-line-options.

See the base function in [ModelComponent.H](#) for info. Actually, here we just keep a copy of `whatToExport` and we will call `exportOptions()` on our [SimulationViewer](#) once we know its type and instantiate it.

Reimplemented from [ModelComponent](#).

Definition at line 77 of file `SimulationViewer.C`.

References `ModelComponent::exportOptions()`, `itsSVtype`, `OPTEXP_CORE`, and `ModelManager::requestOption()`.

### 10.256.3.2 `SharedPtr< SimulationViewer > SimulationViewer- Configurator::getSV () const`

Get the chosen [SimulationViewer](#).

You should call this during `start()` of the [ModelComponent](#) that needs the [SimulationViewer](#). CAUTION: this may be a `SharedPtr(NULL)` if no controller has been chosen.

Definition at line 93 of file `SimulationViewer.C`.

### 10.256.3.3 `bool SimulationViewerConfigurator::setModelParamString (const std::string & name, const std::string & value, const bool recurse = false, const bool warn = true) [protected, virtual]`

Intercept people changing our [ModelParam](#).

See [ModelComponent.H](#); as parsing the command-line or reading a config file sets our name, we'll also here instantiate a controller of the proper type (and export its options)

Reimplemented from [ModelComponent](#).

Definition at line 98 of file `SimulationViewer.C`.

References `ModelComponent::addSubComponent()`, `SharedPtr< SimulationViewer`

>::get(), itsSVtype, ModelComponent::removeSubComponent(), SharedPtr<SimulationViewer >::reset(), and ModelComponent::setModelParamString().

## 10.256.4 Member Data Documentation

### 10.256.4.1 [ModelParam<std::string>](#) [SimulationViewerConfigurator::its-SVtype](#) [protected]

type of viewer

Definition at line 214 of file SimulationViewer.H.

Referenced by exportOptions(), and setModelParamString().

The documentation for this class was generated from the following files:

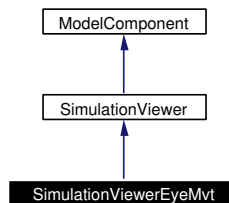
- [SimulationViewer.H](#)
- [SimulationViewer.C](#)

## 10.257 SimulationViewerEyeMvt Class Reference

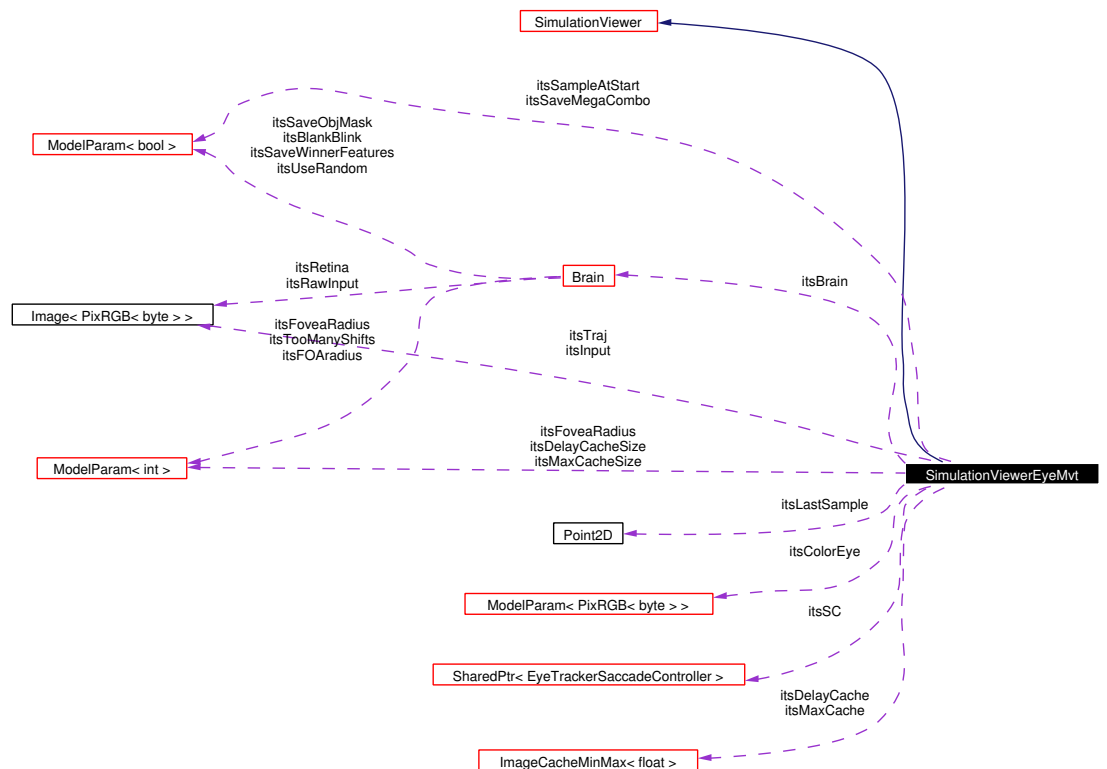
Measure salience at human eye positions.

```
#include <SimulationViewerEyeMvt.H>
```

Inheritance diagram for SimulationViewerEyeMvt:



Collaboration diagram for SimulationViewerEyeMvt:



## Public Member Functions

### Constructors and destructors

- [SimulationViewerEyeMvt](#) ([ModelManager](#) &mgr, const [std::string](#) &descrName="EyeMvt Simulation Viewer", const [std::string](#) &tagName="SimulationViewerEyeMvt")  
*Constructor. See [ModelComponent.H](#).*
- virtual [~SimulationViewerEyeMvt](#) ()  
*Destructor.*
- virtual void [exportOptions](#) (const int whatToExport, const bool recurse=true)  
*Export some of our [ModelParam](#) parameters as command-line-options.*
- virtual void [setBrain](#) ([Brain](#) \*brain)  
*Set our brain.*

### Events that will trigger new visualizations

- virtual void [newInputReady](#) ()  
*Ready to switch to next input frame.*
- virtual void [newInputFrame](#) (const [Image](#)< [PixRGB](#)< [byte](#) > > &input, const double t)  
*Got a new input frame at the specified (current) time.*
- virtual void [channelUpdate](#) ([SharedPtr](#)< [VisualCortex](#) > &vcx)  
*Update with availability of new channel drawings.*
- virtual void [evolve](#) (const double t)  
*Called every simulation time step (time is in seconds).*
- virtual void [attentionShift](#) (const [WTAWinner](#) &win, const [Image](#)< [byte](#) > &foaMask)  
*Update with new covert attention position.*
- virtual void [eyeMovement](#) (const [Point2DT](#) &curreye, const [Image](#)< [byte](#) > &foveaMask)  
*Update with new eye position.*
- virtual void [headMovement](#) (const [Point2DT](#) &currhead, const [Image](#)< [byte](#) > &headMask)  
*Update with new head position.*

- virtual void [eyeSaccade](#) (const bool inSaccade)  
*Update eye saccade status.*
- virtual void [headSaccade](#) (const bool inSaccade)  
*Update head saccade status.*
- virtual void [eyeBlink](#) (const bool inBlink)  
*Update eye blink status.*

### Output of our current internal drawings

- virtual [Image](#)< [PixRGB](#)< byte > > [getTraj](#) (const double t)  
*Get the attention/eye/head trajectory image.*
- virtual void [saveResults](#) (const [SharedPtr](#)< [OutputFrameSeries](#) > &ofs)  
*Save our various results.*

### Protected Member Functions

- virtual void [start1](#) ()  
*Get started.*
- virtual void [stop1](#) ()  
*get stopped*

### Protected Attributes

- [ModelParam](#)< bool > [itsSaveMegaCombo](#)  
*save mega combo?*
- [ModelParam](#)< [PixRGB](#)< byte > > [itsColorEye](#)  
*patch color for human eye mvts*
- [ModelParam](#)< int > [itsFoveaRadius](#)  
*radius of Fovea*
- [ModelParam](#)< int > [itsDelayCacheSize](#)  
*size of our delay cache*
- [ModelParam](#)< int > [itsMaxCacheSize](#)

*size of our max cache*

- [ModelParam](#)< bool > [itsSampleAtStart](#)  
*take samples at start or end of sac*
- [ModelParam](#)< std::string > [itsOutFname](#)  
*Name of a calibrated eye movement file.*

### 10.257.1 Detailed Description

Measure salience at human eye positions.

Definition at line 47 of file SimulationViewerEyeMvt.H.

### 10.257.2 Constructor & Destructor Documentation

**10.257.2.1** [SimulationViewerEyeMvt::SimulationViewerEyeMvt](#)  
([ModelManager](#) & *mgr*, const [std::string](#) & *descrName* =  
"EyeMvt Simulation Viewer", const [std::string](#) & *tagName* =  
"SimulationViewerEyeMvt")

Constructor. See [ModelComponent.H](#).

Definition at line 61 of file SimulationViewerEyeMvt.C.

References [ModelManager::setOptionValString\(\)](#).

**10.257.2.2** [SimulationViewerEyeMvt::~~SimulationViewerEyeMvt](#) ()  
[virtual]

Destructor.

Definition at line 84 of file SimulationViewerEyeMvt.C.

### 10.257.3 Member Function Documentation

**10.257.3.1** void [SimulationViewerEyeMvt::attentionShift](#) (const [WTAwinner](#) &  
*win*, const [Image](#)< [byte](#) > & *foaMask*) [virtual]

Update with new covert attention position.

In [SimulationViewerEyeMvt](#), this is a no-op.

Implements [SimulationViewer](#).

Definition at line 162 of file `SimulationViewerEyeMvt.C`.

**10.257.3.2** `void SimulationViewerEyeMvt::channelUpdate (SharedPtr<VisualCortex> &vcx) [virtual]`

Update with availability of new channel drawings.

In `SimulationViewerEyeMvt`, this is a no-op.

Implements [SimulationViewer](#).

Definition at line 154 of file `SimulationViewerEyeMvt.C`.

**10.257.3.3** `void SimulationViewerEyeMvt::evolve (const double t) [virtual]`

Called every simulation time step (time is in seconds).

In `SimulationViewerEyeMvt`, we evolve the positions of our internal foveas.

Implements [SimulationViewer](#).

Definition at line 158 of file `SimulationViewerEyeMvt.C`.

**10.257.3.4** `void SimulationViewerEyeMvt::exportOptions (const int whatToExport, const bool recurse = true) [virtual]`

Export some of our [ModelParam](#) parameters as command-line-options.

See the base function in [ModelComponent.H](#) for info. Actually, here we just keep a copy of `whatToExport` and we will call `exportOptions()` on our [SaccadeController](#) once we know its type and instantiate it.

Reimplemented from [ModelComponent](#).

Definition at line 88 of file `SimulationViewerEyeMvt.C`.

References `ModelComponent::exportOptions()`, `itsDelayCacheSize`, `itsMaxCacheSize`, `itsSampleAtStart`, `OPTEXP_CORE`, and `ModelManager::requestOption()`.

**10.257.3.5** `void SimulationViewerEyeMvt::eyeBlink (const bool inBlink) [virtual]`

Update eye blink status.

In `SimulationViewerEyeMvt`, this is a no-op.

Implements [SimulationViewer](#).

Definition at line 309 of file `SimulationViewerEyeMvt.C`.



### 10.257.3.6 void SimulationViewerEyeMvt::eyeMovement (const [Point2DT](#) & *curreye*, const [Image](#)< [byte](#) > & *foveaMask*) [virtual]

Update with new eye position.

In SimulationViewerEyeMvt, this is a no-op.

Implements [SimulationViewer](#).

Definition at line 169 of file SimulationViewerEyeMvt.C.

References [Point2D::clampToDims\(\)](#), [Image](#)< [PixRGB](#)< [byte](#) > >::[coordsOk\(\)](#), [Point2D::distance\(\)](#), [Image](#)< [PixRGB](#)< [byte](#) > >::[drawCircle\(\)](#), [Image](#)< [PixRGB](#)< [byte](#) > >::[drawPatch\(\)](#), [SharedPtr](#)< [T](#) >::[dynCastFrom\(\)](#), [ImageCache](#)< [float](#) >::[front\(\)](#), [SharedPtr](#)< [T](#) >::[get\(\)](#), [SharedPtr](#)< [AttentionGuidanceMap](#) >::[get\(\)](#), [Brain::getAGM\(\)](#), [Image](#)< [T](#) >::[getDims\(\)](#), [Image](#)< [PixRGB](#)< [byte](#) > >::[getDims\(\)](#), [Image](#)< [T](#) >::[getHeight\(\)](#), [Image](#)< [T](#) >::[getMinMaxAvg\(\)](#), [ModelComponent::getModelParamVal\(\)](#), [Image](#)< [T](#) >::[getSample\(\)](#), [ModelParam](#)< [bool](#) >::[getVal\(\)](#), [ModelParam](#)< [PixRGB](#)< [byte](#) > >::[getVal\(\)](#), [ModelParam](#)< [int](#) >::[getVal\(\)](#), [Brain::getVC\(\)](#), [Image](#)< [T](#) >::[getWidth\(\)](#), [Point2D::i](#), [itsDelayCacheSize](#), [itsMaxCacheSize](#), [itsSampleAtStart](#), [Point2D::j](#), [LevelSpec::mapLevel\(\)](#), [Point2DT::p](#), [ImageCache](#)< [float](#) >::[push\\_back\(\)](#), [PixRGB](#)< [T](#) >::[setBlue\(\)](#), [PixRGB](#)< [T](#) >::[setGreen\(\)](#), [PixRGB](#)< [T](#) >::[setRed\(\)](#), [ImageCache](#)< [float](#) >::[size\(\)](#), and [uint](#).

### 10.257.3.7 void SimulationViewerEyeMvt::eyeSaccade (const bool *inSaccade*) [virtual]

Update eye saccade status.

In SimulationViewerEyeMvt, this is a no-op.

Implements [SimulationViewer](#).

Definition at line 301 of file SimulationViewerEyeMvt.C.

### 10.257.3.8 [Image](#)< [PixRGB](#)< [byte](#) > > SimulationViewerEyeMvt::getTraj (const double *t*) [virtual]

Get the attention/eye/head trajectory image.

Implements [SimulationViewer](#).

Definition at line 313 of file SimulationViewerEyeMvt.C.

References [concatX\(\)](#), [decXY\(\)](#), [Image](#)< [T](#) >::[drawCircle\(\)](#), [Image](#)< [T](#) >::[drawGrid\(\)](#), [Image](#)< [T](#) >::[drawLine\(\)](#), [SharedPtr](#)< [AttentionGuidanceMap](#) >::[get\(\)](#), [Brain::getAGM\(\)](#), [Image](#)< [PixRGB](#)< [byte](#) > >::[getDims\(\)](#), [ImageCacheMinMax](#)< [float](#) >::[getMax\(\)](#), [ModelParam](#)< [int](#) >::[getVal\(\)](#), [Image](#)< [T](#) >::[getWidth\(\)](#), [Dims::h\(\)](#), [itsMaxCacheSize](#), [lowPass3\(\)](#), [rescale\(\)](#), [ImageCache](#)< [float](#) >::[size\(\)](#), [toRGB\(\)](#), and

Dims::w().

Referenced by saveResults().

**10.257.3.9 void SimulationViewerEyeMvt::headMovement (const [Point2DT](#) & currhead, const [Image](#)< [byte](#) > & headMask) [virtual]**

Update with new head position.

In SimulationViewerEyeMvt, this is a no-op.

Implements [SimulationViewer](#).

Definition at line 296 of file SimulationViewerEyeMvt.C.

**10.257.3.10 void SimulationViewerEyeMvt::headSaccade (const bool inSaccade) [virtual]**

Update head saccade status.

In SimulationViewerEyeMvt, this is a no-op.

Implements [SimulationViewer](#).

Definition at line 305 of file SimulationViewerEyeMvt.C.

**10.257.3.11 void SimulationViewerEyeMvt::newInputFrame (const [Image](#)< [PixRGB](#)< [byte](#) > > & input, const double t) [virtual]**

Got a new input frame at the specified (current) time.

In SimulationViewerEyeMvt, this is a no-op.

Implements [SimulationViewer](#).

Definition at line 144 of file SimulationViewerEyeMvt.C.

**10.257.3.12 void SimulationViewerEyeMvt::newInputReady () [virtual]**

Ready to switch to next input frame.

Implements [SimulationViewer](#).

Definition at line 140 of file SimulationViewerEyeMvt.C.

**10.257.3.13 void SimulationViewerEyeMvt::saveResults (const [SharedPtr](#)< [OutputFrameSeries](#) > & ofs) [virtual]**

Save our various results.

Implements [SimulationViewer](#).

Definition at line 365 of file SimulationViewerEyeMvt.C.

References [getTraj\(\)](#), and [ModelParam< bool >::getVal\(\)](#).

#### 10.257.3.14 void SimulationViewerEyeMvt::setBrain ([Brain](#) \* *brain*) [virtual]

Set our brain.

Reimplemented from [SimulationViewer](#).

Definition at line 109 of file SimulationViewerEyeMvt.C.

#### 10.257.3.15 void SimulationViewerEyeMvt::start1 () [protected, virtual]

Get started.

Reimplemented from [ModelComponent](#).

Definition at line 113 of file SimulationViewerEyeMvt.C.

References [SharedPtr< EyeTrackerSaccadeController >::dynCastFrom\(\)](#), [SharedPtr< EyeTrackerSaccadeController >::get\(\)](#), [Brain::getSC\(\)](#), [ModelParam< int >::getVal\(\)](#), [itsDelayCacheSize](#), [itsMaxCacheSize](#), and [ImageCache< float >::setMaxSize\(\)](#).

#### 10.257.3.16 void SimulationViewerEyeMvt::stop1 () [protected, virtual]

get stopped

Reimplemented from [ModelComponent](#).

Definition at line 134 of file SimulationViewerEyeMvt.C.

### 10.257.4 Member Data Documentation

#### 10.257.4.1 [ModelParam< PixRGB<byte> >](#) [SimulationViewerEyeMvt::its-ColorEye](#) [protected]

patch color for human eye mvts

Definition at line 140 of file SimulationViewerEyeMvt.H.

**10.257.4.2   [ModelParam<int>](#)   [SimulationViewerEyeMvt::itsDelayCacheSize](#)  
[protected]**

size of our delay cache

Definition at line 142 of file SimulationViewerEyeMvt.H.

Referenced by exportOptions(), eyeMovement(), and start1().

**10.257.4.3   [ModelParam<int>](#)   [SimulationViewerEyeMvt::itsFoveaRadius](#)  
[protected]**

radius of Fovea

Definition at line 141 of file SimulationViewerEyeMvt.H.

**10.257.4.4   [ModelParam<int>](#)   [SimulationViewerEyeMvt::itsMaxCacheSize](#)  
[protected]**

size of our max cache

Definition at line 143 of file SimulationViewerEyeMvt.H.

Referenced by exportOptions(), eyeMovement(), getTraj(), and start1().

**10.257.4.5   [ModelParam<std::string>](#)   [SimulationViewerEyeMvt::itsOutFname](#)  
[protected]**

Name of a calibrated eye movement file.

Definition at line 147 of file SimulationViewerEyeMvt.H.

**10.257.4.6   [ModelParam<bool>](#)   [SimulationViewerEyeMvt::itsSampleAtStart](#)  
[protected]**

take samples at start or end of sac

Definition at line 144 of file SimulationViewerEyeMvt.H.

Referenced by exportOptions(), and eyeMovement().

**10.257.4.7   [ModelParam<bool>](#)   [SimulationViewerEyeMvt::itsSaveMega-Combo](#)   [protected]**

save mega combo?

Definition at line 139 of file SimulationViewerEyeMvt.H.

The documentation for this class was generated from the following files:

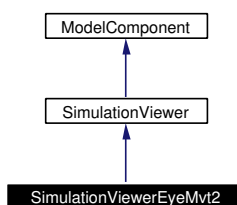
- [SimulationViewerEyeMvt.H](#)
- [SimulationViewerEyeMvt.C](#)

## 10.258 SimulationViewerEyeMvt2 Class Reference

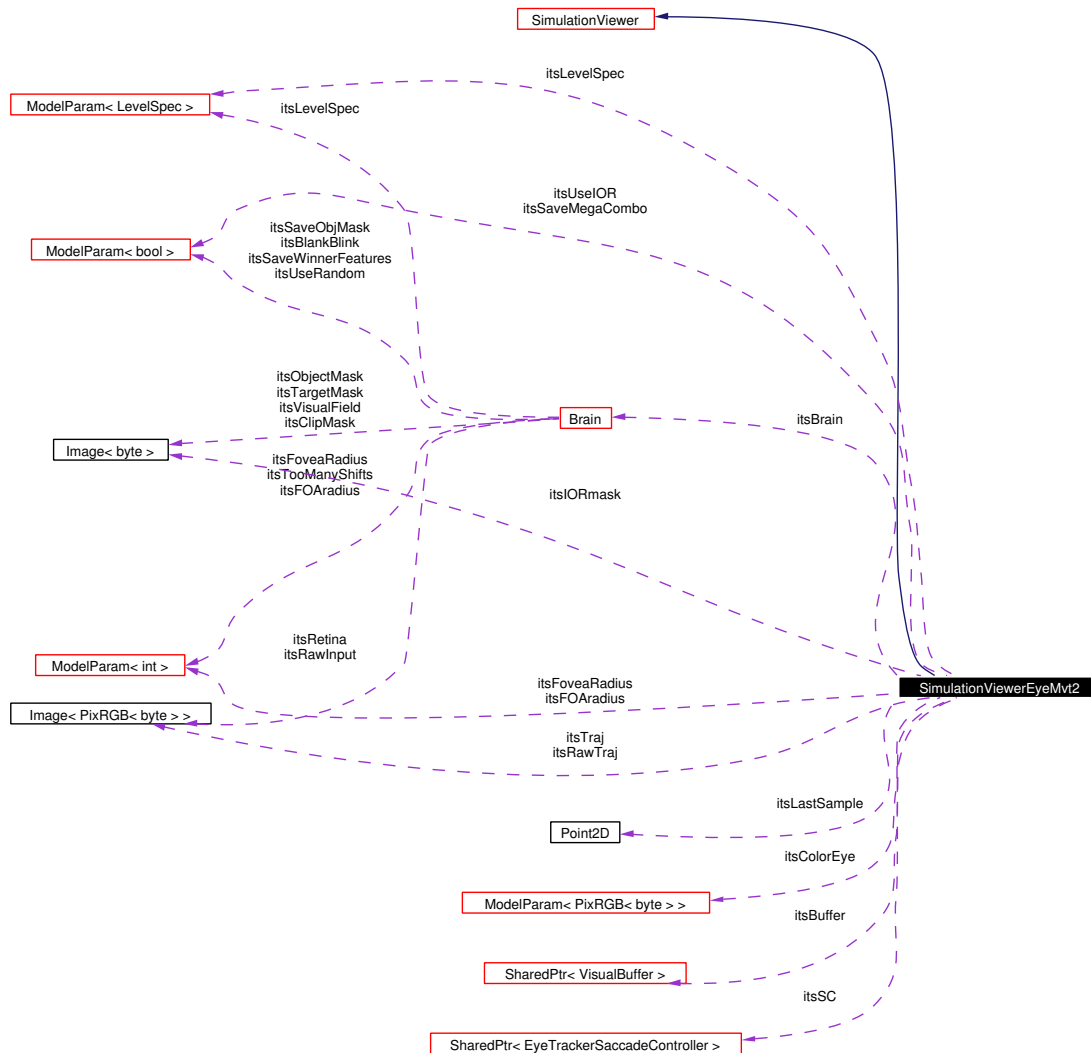
Measure salience at human eye positions.

```
#include <SimulationViewerEyeMvt2.H>
```

Inheritance diagram for SimulationViewerEyeMvt2:



Collaboration diagram for SimulationViewerEyeMvt2:



## Public Member Functions

### Constructors and destructors

- `SimulationViewerEyeMvt2 (ModelManager &mgr, const std::string &descrName="EyeMvt2 Simulation Viewer", const std::string &tagName="SimulationViewerEyeMvt2")`  
*Constructor. See [ModelComponent.H](#).*
- `virtual ~SimulationViewerEyeMvt2 ()`

*Destructor.*

- virtual void [exportOptions](#) (const int whatToExport, const bool recurse=true)  
*Export some of our [ModelParam](#) parameters as command-line-options.*
- virtual void [setBrain](#) (Brain \*brain)  
*Set our brain.*

### Events that will trigger new visualizations

- virtual void [newInputReady](#) ()  
*Ready to switch to next input frame.*
- virtual void [newInputFrame](#) (const [Image](#)< [PixRGB](#)< [byte](#) > > &input, const double t)  
*Got a new input frame at the specified (current) time.*
- virtual void [channelUpdate](#) ([SharedPtr](#)< [VisualCortex](#) > &vcx)  
*Update with availability of new channel drawings.*
- virtual void [evolve](#) (const double t)  
*Called every simulation time step (time is in seconds).*
- virtual void [attentionShift](#) (const [WTAWinner](#) &win, const [Image](#)< [byte](#) > &foaMask)  
*Update with new covert attention position.*
- virtual void [eyeMovement](#) (const [Point2DT](#) &curreye, const [Image](#)< [byte](#) > &foveaMask)  
*Update with new eye position.*
- virtual void [headMovement](#) (const [Point2DT](#) &currhead, const [Image](#)< [byte](#) > &headMask)  
*Update with new head position.*
- virtual void [eyeSaccade](#) (const bool inSaccade)  
*Update eye saccade status.*
- virtual void [headSaccade](#) (const bool inSaccade)  
*Update head saccade status.*
- virtual void [eyeBlink](#) (const bool inBlink)  
*Update eye blink status.*



### Output of our current internal drawings

- virtual [Image](#)< [PixRGB](#)< [byte](#) > > [getTraj](#) (const double t)  
*Get the attention/eye/head trajectory image.*
- virtual void [saveResults](#) (const [SharedPtr](#)< [OutputFrameSeries](#) > &ofs)  
*Save our various results.*

### Protected Member Functions

- virtual void [start1](#) ()  
*Get started.*
- virtual void [stop1](#) ()  
*get stopped*

### Protected Attributes

- [ModelParam](#)< bool > [itsSaveMegaCombo](#)  
*save mega combo?*
- [ModelParam](#)< [PixRGB](#)< [byte](#) > > [itsColorEye](#)  
*patch color for human eye mvts*
- [ModelParam](#)< int > [itsFOARadius](#)  
*radius of FOA*
- [ModelParam](#)< int > [itsFoveaRadius](#)  
*radius of Fovea*
- [ModelParam](#)< [std::string](#) > [itsOutFname](#)  
*Name of output file.*
- [ModelParam](#)< [LevelSpec](#) > [itsLevelSpec](#)  
*levelspec for buffer scale*
- [ModelParam](#)< bool > [itsUseIOR](#)  
*try to use IOR*

### 10.258.1 Detailed Description

Measure salience at human eye positions.

Definition at line 49 of file SimulationViewerEyeMvt2.H.

### 10.258.2 Constructor & Destructor Documentation

**10.258.2.1** `SimulationViewerEyeMvt2::SimulationViewerEyeMvt2(  
    (ModelManager & mgr, const std::string & descrName =  
    "EyeMvt2 Simulation Viewer", const std::string & tagName =  
    "SimulationViewerEyeMvt2"))`

Constructor. See [ModelComponent.H](#).

Definition at line 60 of file SimulationViewerEyeMvt2.C.

References `ModelComponent::addSubComponent()`, and `ModelManager::setOption-ValString()`.

**10.258.2.2** `SimulationViewerEyeMvt2::~~SimulationViewerEyeMvt2 ()  
    [virtual]`

Destructor.

Definition at line 83 of file SimulationViewerEyeMvt2.C.

### 10.258.3 Member Function Documentation

**10.258.3.1** `void SimulationViewerEyeMvt2::attentionShift (const WTAwinner  
    & win, const Image< byte > & foaMask) [virtual]`

Update with new covert attention position.

Will update our internal visual buffer

Implements [SimulationViewer](#).

Definition at line 174 of file SimulationViewerEyeMvt2.C.

References `binaryReverse()`, `WTAwinner::boring`, `Image< T >::clear()`, `Image< PixRGB< byte > >::drawCircle()`, `Image< T >::drawDisk()`, `SharedPtr< T >::dynCast-From()`, `SharedPtr< T >::get()`, `Brain::getAGM()`, `Image< PixRGB< byte > >::getDims()`, `Brain::getSM()`, `Brain::getTRM()`, `ModelParam< int >::getVal()`, `Image< T >::initialized()`, `WTAwinner::p`, `rescale()`, and `Image< T >::resize()`.

### 10.258.3.2 void SimulationViewerEyeMvt2::channelUpdate ([SharedPtr<VisualCortex>](#) & vcx) [virtual]

Update with availability of new channel drawings.

In SimulationViewerEyeMvt2, this is a no-op.

Implements [SimulationViewer](#).

Definition at line 163 of file SimulationViewerEyeMvt2.C.

### 10.258.3.3 void SimulationViewerEyeMvt2::evolve (const double t) [virtual]

Called every simulation time step (time is in seconds).

Implements [SimulationViewer](#).

Definition at line 167 of file SimulationViewerEyeMvt2.C.

### 10.258.3.4 void SimulationViewerEyeMvt2::exportOptions (const int *whatToExport*, const bool *recurse* = true) [virtual]

Export some of our [ModelParam](#) parameters as command-line-options.

See the base function in [ModelComponent.H](#) for info. Actually, here we just keep a copy of whatToExport and we will call [exportOptions\(\)](#) on our [SaccadeController](#) once we know its type and instantiate it.

Reimplemented from [ModelComponent](#).

Definition at line 87 of file SimulationViewerEyeMvt2.C.

References [ModelComponent::exportOptions\(\)](#), [itsUseIOR](#), [OPTEXP\\_CORE](#), and [ModelManager::requestOption\(\)](#).

### 10.258.3.5 void SimulationViewerEyeMvt2::eyeBlink (const bool *inBlink*) [virtual]

Update eye blink status.

In SimulationViewerEyeMvt2, this is a no-op.

Implements [SimulationViewer](#).

Definition at line 338 of file SimulationViewerEyeMvt2.C.

### 10.258.3.6 `void SimulationViewerEyeMvt2::eyeMovement (const Point2DT & curreye, const Image< byte > & foveaMask) [virtual]`

Update with new eye position.

In `SimulationViewerEyeMvt2`, this is a no-op.

Implements [SimulationViewer](#).

Definition at line 212 of file `SimulationViewerEyeMvt2.C`.

References `binaryReverse()`, `Point2D::clampToDims()`, `Image< T >::coordsOk()`, `Point2D::distance()`, `Image< PixRGB< byte > >::drawCircle()`, `Image< PixRGB< byte > >::drawPatch()`, `Image< T >::getDims()`, `Image< T >::getMinMaxAvg()`, `Brain::getRET()`, `Image< T >::getSample()`, `Image< T >::getVal()`, `ModelParam< bool >::getVal()`, `ModelParam< int >::getVal()`, `ModelParam< PixRGB< byte > >::getVal()`, `ModelParam< LevelSpec >::getVal()`, `Dims::h()`, `Point2D::i`, `itsUseIOR`, `Point2D::j`, `Image< T >::lowThresh()`, `LevelSpec::mapLevel()`, `Image< T >::normalize()`, `Point2DT::p`, `segmentObjectClean()`, `PixRGB< T >::setBlue()`, `PixRGB< T >::setGreen()`, `PixRGB< T >::setRed()`, `Dims::w()`, and `ZEROS`.

### 10.258.3.7 `void SimulationViewerEyeMvt2::eyeSaccade (const bool inSaccade) [virtual]`

Update eye saccade status.

In `SimulationViewerEyeMvt2`, this is a no-op.

Implements [SimulationViewer](#).

Definition at line 330 of file `SimulationViewerEyeMvt2.C`.

### 10.258.3.8 `Image< PixRGB< byte > > SimulationViewerEyeMvt2::getTraj (const double t) [virtual]`

Get the attention/eye/head trajectory image.

Implements [SimulationViewer](#).

Definition at line 342 of file `SimulationViewerEyeMvt2.C`.

References `Image< T >::clear()`, `concatX()`, `concatY()`, `contour2D8()`, `Image< T >::drawCircle()`, `Image< T >::drawDisk()`, `Image< T >::drawLine()`, `Image< byte >::freeMem()`, `Brain::getAGM()`, `Image< PixRGB< byte > >::getDims()`, `Image< PixRGB< byte > >::getHeight()`, `Image< T >::getHeight()`, `Image< T >::getVal()`, `ModelParam< LevelSpec >::getVal()`, `Image< PixRGB< byte > >::getWidth()`, `Image< T >::getWidth()`, `Dims::h()`, `Point2D::i`, `Image< byte >::initialized()`, `Point2D::j`, `LevelSpec::mapLevel()`, `NO_INIT`, `Image< T >::paste()`, `rescale()`, `Image< T >::resize()`, `toRGB()`, and `Dims::w()`.

Referenced by `saveResults()`.

**10.258.3.9** `void SimulationViewerEyeMvt2::headMovement (const Point2DT & currhead, const Image< byte > & headMask)` [virtual]

Update with new head position.

In `SimulationViewerEyeMvt2`, this is a no-op.

Implements [SimulationViewer](#).

Definition at line 325 of file `SimulationViewerEyeMvt2.C`.

**10.258.3.10** `void SimulationViewerEyeMvt2::headSaccade (const bool inSaccade)` [virtual]

Update head saccade status.

In `SimulationViewerEyeMvt2`, this is a no-op.

Implements [SimulationViewer](#).

Definition at line 334 of file `SimulationViewerEyeMvt2.C`.

**10.258.3.11** `void SimulationViewerEyeMvt2::newInputFrame (const Image< PixRGB< byte > > & input, const double t)` [virtual]

Got a new input frame at the specified (current) time.

In `SimulationViewerEyeMvt2`, this is a no-op.

Implements [SimulationViewer](#).

Definition at line 141 of file `SimulationViewerEyeMvt2.C`.

References `Brain::getRET()`, `Brain::getSM()`, and `ZEROS`.

**10.258.3.12** `void SimulationViewerEyeMvt2::newInputReady ()` [virtual]

Ready to switch to next input frame.

Implements [SimulationViewer](#).

Definition at line 137 of file `SimulationViewerEyeMvt2.C`.

**10.258.3.13** `void SimulationViewerEyeMvt2::saveResults (const SharedPtr< OutputFrameSeries > & ofs)` [virtual]

Save our various results.

Implements [SimulationViewer](#).

Definition at line 428 of file SimulationViewerEyeMvt2.C.

References [getTraj\(\)](#), and [ModelParam< bool >::getVal\(\)](#).

#### 10.258.3.14 **void SimulationViewerEyeMvt2::setBrain ([Brain](#) \* brain)** [virtual]

Set our brain.

Reimplemented from [SimulationViewer](#).

Definition at line 108 of file SimulationViewerEyeMvt2.C.

#### 10.258.3.15 **void SimulationViewerEyeMvt2::start1 ()** [protected, virtual]

Get started.

Reimplemented from [ModelComponent](#).

Definition at line 112 of file SimulationViewerEyeMvt2.C.

References [SharedPtr< EyeTrackerSaccadeController >::dynCastFrom\(\)](#), [SharedPtr< Retina >::get\(\)](#), [SharedPtr< EyeTrackerSaccadeController >::get\(\)](#), [Brain::getRET\(\)](#), and [Brain::getSC\(\)](#).

#### 10.258.3.16 **void SimulationViewerEyeMvt2::stop1 ()** [protected, virtual]

get stopped

Reimplemented from [ModelComponent](#).

Definition at line 131 of file SimulationViewerEyeMvt2.C.

### 10.258.4 Member Data Documentation

#### 10.258.4.1 **[ModelParam< PixRGB<byte> > SimulationViewerEyeMvt2::its-ColorEye](#)** [protected]

patch color for human eye mvts

Definition at line 140 of file SimulationViewerEyeMvt2.H.

**10.258.4.2** [ModelParam<int> SimulationViewerEyeMvt2::itsFOAradius](#)  
[protected]

radius of FOA

Definition at line 141 of file SimulationViewerEyeMvt2.H.

**10.258.4.3** [ModelParam<int> SimulationViewerEyeMvt2::itsFoveaRadius](#)  
[protected]

radius of Fovea

Definition at line 142 of file SimulationViewerEyeMvt2.H.

**10.258.4.4** [ModelParam<LevelSpec> SimulationViewerEyeMvt2::itsLevelSpec](#)  
[protected]

levelspec for buffer scale

Definition at line 144 of file SimulationViewerEyeMvt2.H.

**10.258.4.5** [ModelParam<std::string> SimulationViewerEyeMvt2::itsOut-Fname](#) [protected]

Name of output file.

Definition at line 143 of file SimulationViewerEyeMvt2.H.

**10.258.4.6** [ModelParam<bool> SimulationViewerEyeMvt2::itsSaveMega-Combo](#) [protected]

save mega combo?

Definition at line 139 of file SimulationViewerEyeMvt2.H.

**10.258.4.7** [ModelParam<bool> SimulationViewerEyeMvt2::itsUseIOR](#)  
[protected]

try to use IOR

Definition at line 145 of file SimulationViewerEyeMvt2.H.

Referenced by exportOptions(), and eyeMovement().

The documentation for this class was generated from the following files:

- [SimulationViewerEyeMvt2.H](#)
- [SimulationViewerEyeMvt2.C](#)

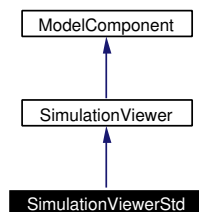


## 10.259 SimulationViewerStd Class Reference

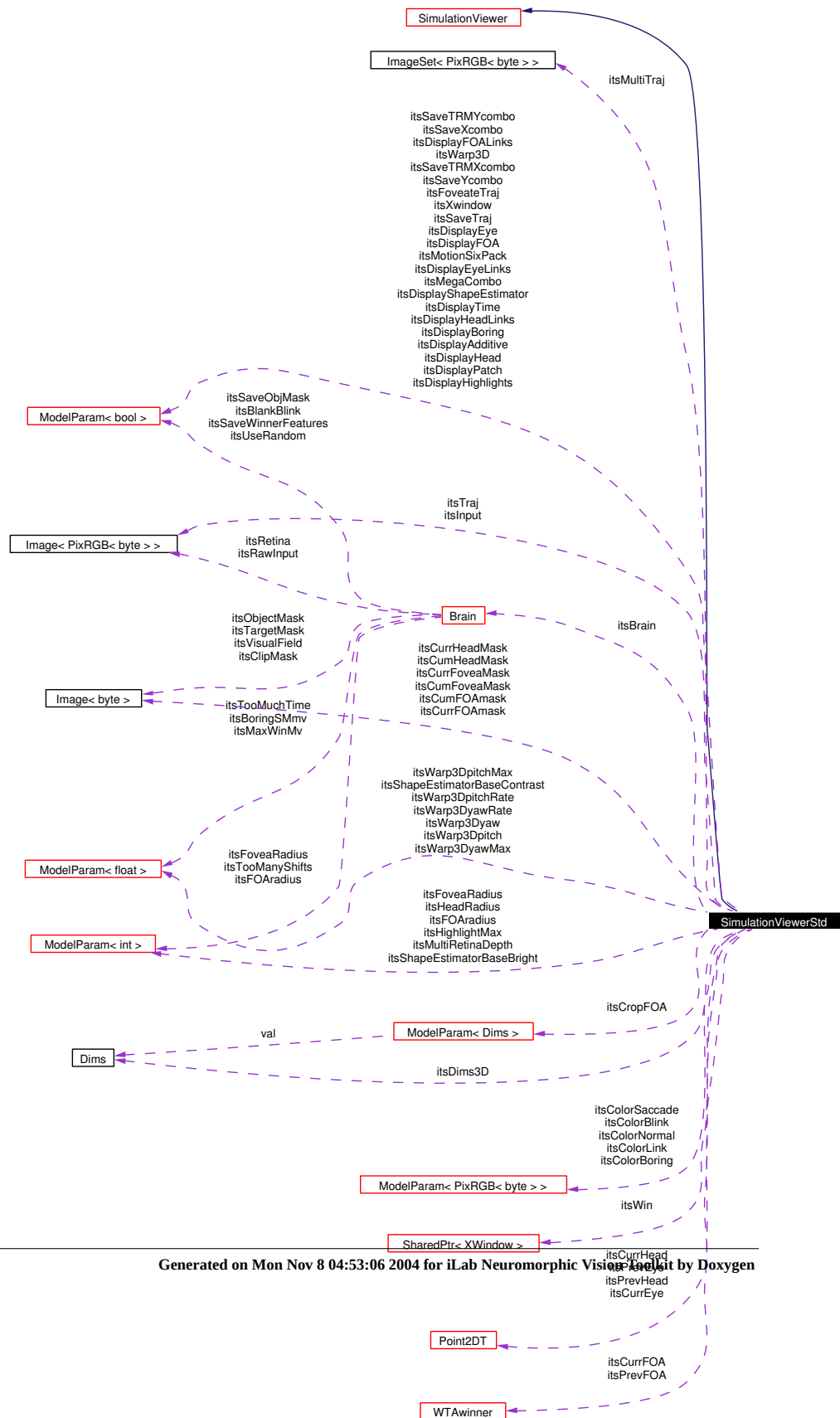
This class provides a standard set of visualization tools.

```
#include <SimulationViewerStd.H>
```

Inheritance diagram for SimulationViewerStd:



Collaboration diagram for SimulationViewerStd:



## Public Member Functions

### Constructors and destructors

- [SimulationViewerStd](#) ([ModelManager](#) &mgr, const [std::string](#) &descrName="Simulation Viewer Std", const [std::string](#) &tagName="SimulationViewerStd")  
*Constructor. See [ModelComponent.H](#).*
- virtual [~SimulationViewerStd](#) ()  
*Destructor.*
- virtual void [exportOptions](#) (const int whatToExport, const bool recurse=true)  
*Export some of our [ModelParam](#) parameters as command-line-options.*
- virtual void [reset](#) (const bool recurse=true)  
*Reset SimulationViewerStd.*
- virtual void [setBrain](#) ([Brain](#) \*brain)  
*Set our brain.*

### Events that will trigger new drawings

- void [newInputReady](#) ()  
*Ready to switch to next input frame.*
- void [newInputFrame](#) (const [Image](#)< [PixRGB](#)< [byte](#) > > &input, const double t)  
*Got a new input frame.*
- void [channelUpdate](#) ([SharedPtr](#)< [VisualCortex](#) > &vcx)  
*Update with availability of new channel drawings.*
- void [evolve](#) (const double t)  
*Called every simulation time step (time is in seconds).*
- void [attentionShift](#) (const [WTAwinner](#) &win, const [Image](#)< [byte](#) > &foveaMask)  
*Update with new covert attention position.*
- void [eyeMovement](#) (const [Point2DT](#) &curreye, const [Image](#)< [byte](#) > &foveaMask)  
*Update with new eye position.*

- void [headMovement](#) (const [Point2DT](#) &currhead, const [Image](#)< [byte](#) > &headMask)  
*Update with new head position.*
- void [eyeSaccade](#) (const bool inSaccade)  
*Update eye saccade status.*
- void [headSaccade](#) (const bool inSaccade)  
*Update head saccade status.*
- void [eyeBlink](#) (const bool inBlink)  
*Update eye blink status.*

### Output of our current internal drawings

- [Image](#)< [PixRGB](#)< [byte](#) > > [getTraj](#) (const double t)  
*Get the attention/eye/head trajectory image.*
- void [saveResults](#) (const [SharedPtr](#)< [OutputFrameSeries](#) > &ofs)  
*Save our various results.*

## Protected Member Functions

### Derived classes should overload those to fit their cosmetic needs

- void [drawFOA](#) ()  
*Draw the focus of covert attention.*
- void [linkFOAs](#) ()  
*Link previous focus of attention to current one.*
- void [drawEye](#) ()  
*Draw current eye position.*
- void [linkEyes](#) ()  
*Link previous eye position to current one.*
- void [drawHead](#) ()  
*Draw current head position.*
- void [linkHeads](#) ()  
*Link previous head position to current one.*
- void [drawTime](#) ([Image](#)< [PixRGB](#)< [byte](#) > > &image)

*Draw current time onto given image.*

- void [drawHighlights](#) ()  
*Draw highlights around a given position.*
- [Image](#)< [PixRGB](#)< [byte](#) > > [drawMegaCombo](#) ()  
*Draw mega combo for X display.*
- [Image](#)< [PixRGB](#)< [byte](#) > > [drawMotionSixPack](#) ()  
*Draw motion sixpack.*
- void [drawMaskOutline](#) ([Image](#)< [PixRGB](#)< [byte](#) > > &traj, const [Image](#)< [byte](#) > mask, const [PixRGB](#)< [byte](#) > &col, const int thick, const [Point2D](#) &pos, const int radius)  
*Draw outline of a mask (used by drawFOA, drawEye, etc).*

## Protected Attributes

- [ModelParam](#)< bool > **itsSaveTraj**
- [ModelParam](#)< bool > **itsSaveXcombo**
- [ModelParam](#)< bool > **itsSaveYcombo**
- [ModelParam](#)< bool > **itsSaveTRMXcombo**
- [ModelParam](#)< bool > **itsSaveTRMYcombo**
- [ModelParam](#)< bool > **itsWarp3D**
- [ModelParam](#)< bool > **itsMegaCombo**
- [ModelParam](#)< bool > **itsMotionSixPack**
- [ModelParam](#)< [Dims](#) > **itsCropFOA**
- [ModelParam](#)< bool > **itsXwindow**
- [ModelParam](#)< bool > **itsFoveateTraj**
- [ModelParam](#)< bool > **itsDisplayFOA**
- [ModelParam](#)< bool > **itsDisplayPatch**
- [ModelParam](#)< bool > **itsDisplayFOALinks**
- [ModelParam](#)< bool > **itsDisplayEye**
- [ModelParam](#)< bool > **itsDisplayEyeLinks**
- [ModelParam](#)< bool > **itsDisplayHead**
- [ModelParam](#)< bool > **itsDisplayHeadLinks**
- [ModelParam](#)< bool > **itsDisplayTime**
- [ModelParam](#)< bool > **itsDisplayAdditive**
- [ModelParam](#)< bool > **itsDisplayHighlights**
- [ModelParam](#)< bool > **itsDisplayBoring**
- [ModelParam](#)< bool > **itsDisplayShapeEstimator**
- [ModelParam](#)< [PixRGB](#)< [byte](#) > > **itsColorNormal**
- [ModelParam](#)< [PixRGB](#)< [byte](#) > > **itsColorBoring**

*This gets subtracted (with clamping) from the current color if boring.*

- [ModelParam](#)< [PixRGB](#)< [byte](#) > > [itsColorBlink](#)
- [ModelParam](#)< [PixRGB](#)< [byte](#) > > [itsColorSaccade](#)
- [ModelParam](#)< [PixRGB](#)< [byte](#) > > [itsColorLink](#)
- [ModelParam](#)< [int](#) > [itsHighlightMax](#)
- [ModelParam](#)< [float](#) > [itsShapeEstimatorBaseContrast](#)
- [ModelParam](#)< [int](#) > [itsShapeEstimatorBaseBright](#)
- [ModelParam](#)< [float](#) > [itsWarp3Dpitch](#)
- [ModelParam](#)< [float](#) > [itsWarp3Dyaw](#)
- [ModelParam](#)< [float](#) > [itsWarp3DpitchRate](#)
- [ModelParam](#)< [float](#) > [itsWarp3DyawRate](#)
- [ModelParam](#)< [float](#) > [itsWarp3DpitchMax](#)
- [ModelParam](#)< [float](#) > [itsWarp3DyawMax](#)
- [ModelParam](#)< [int](#) > [itsFOAradius](#)
- [ModelParam](#)< [int](#) > [itsFoveaRadius](#)
- [ModelParam](#)< [int](#) > [itsHeadRadius](#)
- [ModelParam](#)< [int](#) > [itsMultiRetinaDepth](#)

### 10.259.1 Detailed Description

This class provides a standard set of visualization tools.

Its typical usage is with a model that generates shifts of attention, eye and head movements. This class will handle visualizing those events. There are three ways of customizing it to your specific needs: 1) by changing the values of some of its internal [ModelParam](#) parameters, either via the command-line or via a config file for those that do not have command-line options; 2) by overloading (in a derived class) the low-level drawing functions in the protected section of the class definition; 3) by overloading the trigger functions in the public section of the class definition.

Definition at line 61 of file [SimulationViewerStd.H](#).

### 10.259.2 Constructor & Destructor Documentation

#### 10.259.2.1 [SimulationViewerStd::SimulationViewerStd](#) ([ModelManager](#) & *mgr*, const [std::string](#) & *descrName* = "Simulation Viewer Std", const [std::string](#) & *tagName* = "SimulationViewerStd")

Constructor. See [ModelComponent.H](#).

#### Parameters:

- mgr* our [ModelManager](#) (see [ModelManager.H](#))
- descrName* descriptive name for human useag

*tagName* name for [ParamMap](#) usage

Definition at line 59 of file SimulationViewerStd.C.

#### 10.259.2.2 SimulationViewerStd::~~SimulationViewerStd () [virtual]

Destructor.

Definition at line 132 of file SimulationViewerStd.C.

### 10.259.3 Member Function Documentation

#### 10.259.3.1 void SimulationViewerStd::attentionShift (const [WTAWinner](#) & win, const [Image](#)< [byte](#) > & foaMask) [virtual]

Update with new covert attention position.

Implements [SimulationViewer](#).

Definition at line 276 of file SimulationViewerStd.C.

References [Image](#)< [byte](#) >::drawDisk(), drawFOA(), [Image](#)< [PixRGB](#)< [byte](#) > >::getDims(), [ModelParam](#)< [bool](#) >::getVal(), [ModelParam](#)< [int](#) >::getVal(), [Image](#)< [PixRGB](#)< [byte](#) > >::initialized(), [Image](#)< [byte](#) >::initialized(), [Image](#)< [T](#) >::initialized(), [Image](#)< [byte](#) >::isSameSize(), [Image](#)< [T](#) >::isSameSize(), linkFOAs(), [Image](#)< [byte](#) >::resize(), and takeMax().

#### 10.259.3.2 void SimulationViewerStd::channelUpdate ([SharedPtr](#)< [VisualCortex](#) > & vcx) [virtual]

Update with availability of new channel drawings.

Implements [SimulationViewer](#).

Definition at line 245 of file SimulationViewerStd.C.

References [SharedPtr](#)< [T](#) >::get(), and [Image](#)< [PixRGB](#)< [byte](#) > >::initialized().

#### 10.259.3.3 void SimulationViewerStd::drawEye () [protected]

Draw current eye position.

Definition at line 591 of file SimulationViewerStd.C.

References [WTAWinner](#)::boring, drawMaskOutline(), [Image](#)< [PixRGB](#)< [byte](#) > >::drawRect(), [Image](#)< [PixRGB](#)< [byte](#) > >::getDims(), [ModelParam](#)< [int](#) >::getVal(), [ModelParam](#)< [bool](#) >::getVal(), [ModelParam](#)< [PixRGB](#)< [byte](#) > >::getVal(), [Point2D](#)::i, [Point2DT](#)::isValid(), itsColorBoring, [Point2D](#)::j, and [Point2DT](#)::p.

Referenced by eyeMovement(), and getTraj().

#### 10.259.3.4 void SimulationViewerStd::drawFOA () [protected]

Draw the focus of covert attention.

Definition at line 555 of file SimulationViewerStd.C.

References WTAwinner::boring, drawMaskOutline(), Image< PixRGB< byte > >::drawPatch(), ModelParam< int >::getVal(), ModelParam< bool >::getVal(), ModelParam< PixRGB< byte > >::getVal(), WTAwinner::isValid(), itsColorBoring, and WTAwinner::p.

Referenced by attentionShift(), and getTraj().

#### 10.259.3.5 void SimulationViewerStd::drawHead () [protected]

Draw current head position.

Definition at line 630 of file SimulationViewerStd.C.

References WTAwinner::boring, drawMaskOutline(), Image< PixRGB< byte > >::drawRect(), Image< PixRGB< byte > >::getDims(), ModelParam< int >::getVal(), ModelParam< bool >::getVal(), ModelParam< PixRGB< byte > >::getVal(), Point2D::i, Point2DT::isValid(), itsColorBoring, Point2D::j, and Point2DT::p.

Referenced by getTraj(), and headMovement().

#### 10.259.3.6 void SimulationViewerStd::drawHighlights () [protected]

Draw highlights around a given position.

#### 10.259.3.7 void SimulationViewerStd::drawMaskOutline (Image< PixRGB< byte > > & traj, const Image< byte > mask, const PixRGB< byte > & col, const int thick, const Point2D & pos, const int radius) [protected]

Draw outline of a mask (used by drawFOA, drawEye, etc).

if the mask is initialized, revert to drawing a circle centered at "pos" an of radius "radius"

Definition at line 830 of file SimulationViewerStd.C.

References contour2D8(), Image< T >::getVal(), Point2D::i, Point2D::j, and Image< T >::lowThresh().

Referenced by drawEye(), drawFOA(), and drawHead().



### 10.259.3.8 **Image< PixRGB< byte > > SimulationViewerStd::drawMegaCombo ()** [protected]

Draw mega combo for X display.

Definition at line 674 of file SimulationViewerStd.C.

References Image< T >::clamp(), Image< T >::drawRect(), SharedPtr< T >::dynCastFrom(), Image< T >::embed(), SharedPtr< T >::get(), SharedPtr< VisualCortex >::get(), SharedPtr< TaskRelevanceMap >::get(), SharedPtr< SaliencyMap >::get(), Image< PixRGB< byte > >::getHeight(), Brain::getSM(), Brain::getTRM(), Brain::getVC(), Image< PixRGB< byte > >::getWidth(), makeRGB(), Rectangle::modify(), NO\_INIT, Image< T >::normalize(), rescale(), Image< T >::resize(), Image< T >::writeText(), XWW, and ZEROS.

Referenced by getTraj().

### 10.259.3.9 **Image< PixRGB< byte > > SimulationViewerStd::drawMotionSixPack ()** [protected]

Draw motion sixpack.

Definition at line 784 of file SimulationViewerStd.C.

References concatX(), concatY(), Image< T >::drawArrow(), Image< T >::drawGrid(), SharedPtr< T >::dynCastFrom(), SharedPtr< T >::get(), SharedPtr< VisualCortex >::get(), Image< PixRGB< byte > >::getDims(), Brain::getVC(), Dims::h(), luminance(), scaleBlock(), and Dims::w().

Referenced by getTraj().

### 10.259.3.10 **void SimulationViewerStd::drawTime (Image< PixRGB< byte > > & image)** [protected]

Draw current time onto given image.

Definition at line 667 of file SimulationViewerStd.C.

References image, and Image< PixRGB< float > >::writeText().

Referenced by getTraj().

### 10.259.3.11 **void SimulationViewerStd::evolve (const double t)** [virtual]

Called every simulation time step (time is in seconds).

Implements [SimulationViewer](#).

Definition at line 252 of file SimulationViewerStd.C.

References `ModelParam< float >::getVal()`, `ModelParam< bool >::getVal()`, and `ModelParam< float >::setVal()`.

**10.259.3.12 `void SimulationViewerStd::exportOptions (const int whatToExport, const bool recurse = true) [virtual]`**

Export some of our [ModelParam](#) parameters as command-line-options.

See the base function in [ModelComponent.H](#) for info.

Reimplemented from [ModelComponent](#).

Definition at line 136 of file `SimulationViewerStd.C`.

References `ModelComponent::exportOptions()`, `OPTEXP_CORE`, and `ModelManager::requestOption()`.

**10.259.3.13 `void SimulationViewerStd::eyeBlink (const bool inBlink) [virtual]`**

Update eye blink status.

Implements [SimulationViewer](#).

Definition at line 398 of file `SimulationViewerStd.C`.

**10.259.3.14 `void SimulationViewerStd::eyeMovement (const Point2DT & curreye, const Image< byte > & foveaMask) [virtual]`**

Update with new eye position.

Implements [SimulationViewer](#).

Definition at line 314 of file `SimulationViewerStd.C`.

References `Image< byte >::drawDisk()`, `drawEye()`, `Image< PixRGB< byte > >::getDims()`, `ModelParam< bool >::getVal()`, `ModelParam< int >::getVal()`, `Image< PixRGB< byte > >::initialized()`, `Image< byte >::initialized()`, `Image< T >::initialized()`, `Image< byte >::isSameSize()`, `Image< T >::isSameSize()`, `linkEyes()`, `Image< byte >::resize()`, and `takeMax()`.

**10.259.3.15 `void SimulationViewerStd::eyeSaccade (const bool inSaccade) [virtual]`**

Update eye saccade status.

Implements [SimulationViewer](#).

Definition at line 390 of file `SimulationViewerStd.C`.

### 10.259.3.16 **Image< PixRGB< byte > > SimulationViewerStd::getTraj (const double t) [virtual]**

Get the attention/eye/head trajectory image.

Implements [SimulationViewer](#).

Definition at line 402 of file SimulationViewerStd.C.

References [contrastModulate\(\)](#), [crop\(\)](#), [drawEye\(\)](#), [drawFOA\(\)](#), [Image< T >::drawGrid\(\)](#), [drawHead\(\)](#), [drawMegaCombo\(\)](#), [drawMotionSixPack\(\)](#), [Image< T >::drawRect\(\)](#), [drawTime\(\)](#), [Image< PixRGB< byte > >::getDims\(\)](#), [Image< T >::getHeight\(\)](#), [Brain::getSM\(\)](#), [Brain::getTRM\(\)](#), [ModelParam< PixRGB< byte > >::getVal\(\)](#), [ModelParam< Dims >::getVal\(\)](#), [ModelParam< int >::getVal\(\)](#), [ModelParam< float >::getVal\(\)](#), [ModelParam< bool >::getVal\(\)](#), [Image< T >::getWidth\(\)](#), [Dims::h\(\)](#), [Point2D::i](#), [Image< PixRGB< byte > >::initialized\(\)](#), [Dims::isNonEmpty\(\)](#), [Point2D::j](#), [Image< T >::normalize\(\)](#), [WTAwinner::p](#), [rescale\(\)](#), [toRGB\(\)](#), and [Dims::w\(\)](#).

Referenced by [saveResults\(\)](#).

### 10.259.3.17 **void SimulationViewerStd::headMovement (const [Point2DT](#) & currhead, const [Image< byte >](#) & headMask) [virtual]**

Update with new head position.

Implements [SimulationViewer](#).

Definition at line 352 of file SimulationViewerStd.C.

References [Image< byte >::drawDisk\(\)](#), [drawHead\(\)](#), [Image< PixRGB< byte > >::getDims\(\)](#), [ModelParam< bool >::getVal\(\)](#), [ModelParam< int >::getVal\(\)](#), [Image< PixRGB< byte > >::initialized\(\)](#), [Image< byte >::initialized\(\)](#), [Image< T >::initialized\(\)](#), [Image< byte >::isSameSize\(\)](#), [Image< T >::isSameSize\(\)](#), [linkHeads\(\)](#), [Image< byte >::resize\(\)](#), and [takeMax\(\)](#).

### 10.259.3.18 **void SimulationViewerStd::headSaccade (const bool inSaccade) [virtual]**

Update head saccade status.

Implements [SimulationViewer](#).

Definition at line 394 of file SimulationViewerStd.C.

### 10.259.3.19 **void SimulationViewerStd::linkEyes () [protected]**

Link previous eye position to current one.

Definition at line 617 of file `SimulationViewerStd.C`.

References `Image< PixRGB< byte > >::drawLine()`, `ModelParam< PixRGB< byte > >::getVal()`, `Point2DT::isValid()`, and `Point2DT::p`.

Referenced by `eyeMovement()`.

#### 10.259.3.20 `void SimulationViewerStd::linkFOAs ()` [protected]

Link previous focus of attention to current one.

Definition at line 576 of file `SimulationViewerStd.C`.

References `Point2D::distance()`, `Image< PixRGB< byte > >::drawArrow()`, `ModelParam< PixRGB< byte > >::getVal()`, `WTWinner::isValid()`, and `WTWinner::p`.

Referenced by `attentionShift()`.

#### 10.259.3.21 `void SimulationViewerStd::linkHeads ()` [protected]

Link previous head position to current one.

Definition at line 654 of file `SimulationViewerStd.C`.

References `Image< PixRGB< byte > >::drawLine()`, `ModelParam< PixRGB< byte > >::getVal()`, `Point2DT::isValid()`, and `Point2DT::p`.

Referenced by `headMovement()`.

#### 10.259.3.22 `void SimulationViewerStd::newInputFrame (const Image< PixRGB< byte > > & input, const double t)` [virtual]

Got a new input frame.

Implements [SimulationViewer](#).

Definition at line 223 of file `SimulationViewerStd.C`.

References `buildPyrGaussian()`, `ModelParam< int >::getVal()`, and `ModelParam< bool >::getVal()`.

#### 10.259.3.23 `void SimulationViewerStd::newInputReady ()` [virtual]

Ready to switch to next input frame.

Each of these will trigger a sequence of calls to the lower-level functions in the protected section of this class definition. These sequences will depend on the current settings of our [ModelParam](#) parameters.

Implements [SimulationViewer](#).

Definition at line 219 of file SimulationViewerStd.C.

#### 10.259.3.24 void SimulationViewerStd::reset (const bool *recurse* = true) [virtual]

Reset SimulationViewerStd.

See the base function in [ModelComponent.H](#) for info.

Reimplemented from [ModelComponent](#).

Definition at line 182 of file SimulationViewerStd.C.

References [Image< byte >::freeMem\(\)](#), [Image< PixRGB< byte > >::freeMem\(\)](#), [ModelComponent::reset\(\)](#), [SharedPtr< XWindow >::reset\(\)](#), and [ImageSet< Pix-  
RGB< byte > >::reset\(\)](#).

#### 10.259.3.25 void SimulationViewerStd::saveResults (const [SharedPtr< OutputFrameSeries > & ofs](#)) [virtual]

Save our various results.

Implements [SimulationViewer](#).

Definition at line 517 of file SimulationViewerStd.C.

References [SharedPtr< XWindow >::get\(\)](#), [Image< T >::getHeight\(\)](#), [getTraj\(\)](#), [ModelParam< bool >::getVal\(\)](#), [Image< T >::getWidth\(\)](#), and [makeShared\(\)](#).

#### 10.259.3.26 void SimulationViewerStd::setBrain ([Brain](#) \* *brain*) [virtual]

Set our brain.

Some drawings, in particular those involving the saliency map (like x-combo, warp3d, etc) require a brain

Reimplemented from [SimulationViewer](#).

Definition at line 215 of file SimulationViewerStd.C.

### 10.259.4 Member Data Documentation

#### 10.259.4.1 [ModelParam< PixRGB<byte> > SimulationViewerStd::itsColor-Boring](#) [protected]

This gets subtracted (with clamping) from the current color if boring.

Definition at line 220 of file SimulationViewerStd.H.

Referenced by `drawEye()`, `drawFOA()`, and `drawHead()`.

The documentation for this class was generated from the following files:

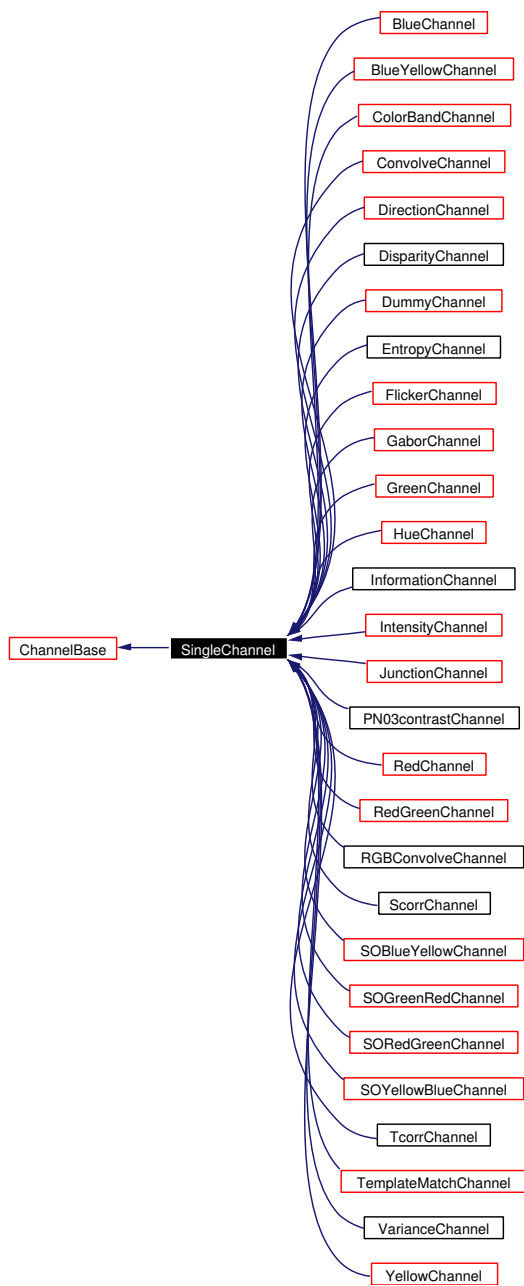
- [SimulationViewerStd.H](#)
- [SimulationViewerStd.C](#)

## 10.260 SingleChannel Class Reference

SingleChannel represents a single stream of processing.

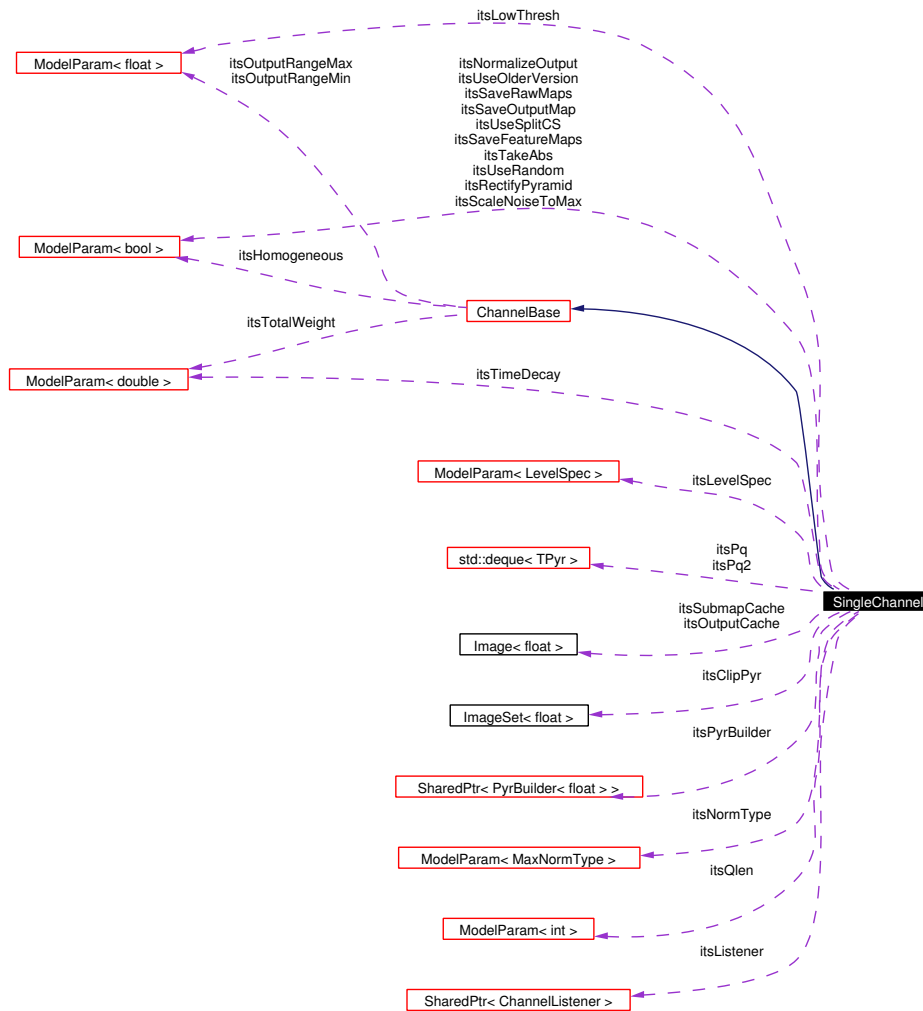
```
#include <SingleChannel.H>
```

Inheritance diagram for SingleChannel:



Collaboration diagram for `SingleChannel`:





## Public Member Functions

- **SingleChannel** (**ModelManager** &mgr, const **std::string** &descrName, const **std::string** &tagName, const **VisualFeature** vs, **SharedPtr**< **PyrBuilder**< float > > pyr)  
*Constructor. See [ChannelBase.H](#).*
- virtual **~SingleChannel** ()  
*destructor*

- virtual void [exportOptions](#) (const int whatToExport, const bool recurse=true)  
*Export some of our [ModelParam](#) parameters as command-line-options.*
- virtual void [reset](#) (const bool recurse=true)  
*Reset [SingleChannel](#).*
- virtual void [inputPyramid](#) (const [ImageSet](#)< float > &pyramid, const double t, const [Image](#)< byte > &clipMask=[Image](#)< byte >())  
*Send an already computed pyramid as input to the channel.*
- virtual [ImageSet](#)< float > [computePyramid](#) (const [Image](#)< float > &bwimg)  
*Computes the channel's pyramid but doesn't store it.*
- virtual void [readFrom](#) (const [ParamMap](#) &pmap)  
*Read params from the [ParamMap](#).*
- virtual void [writeTo](#) ([ParamMap](#) &pmap) const  
*Write params to the [ParamMap](#).*
- virtual void [setListener](#) ([SharedPtr](#)< [ChannelListener](#) > listener)  
*Set a [ChannelListener](#) for this channel.*
- virtual [uint](#) [numChans](#) () const  
*number of subchannels for [ComplexChannel](#), 0 for [SingleChannel](#)*
- virtual void [cleanCoeffs](#) (const double eps)  
*Set any coefficients less than eps to zero.*
- virtual void [clampCoeffs](#) (const double cmin, const double cmax)  
*Clamp the weighting coefficients to the specified range.*
- virtual double [absSumCoeffs](#) () const  
*Return the sum of the absolute values of the weighting coefficients.*
- virtual void [normalizeCoeffs](#) (const double div)  
*Normalize the weighting coefficients by dividing by div.*
- virtual void [setCoeff](#) (const [uint](#) cntr, const [uint](#) surr, const double val)  
*Set the weighting coefficient for the given center/surround level.*
- virtual void [setMean](#) (const [uint](#) cntr, const [uint](#) surr, const double val)
- virtual void [setSigma](#) (const [uint](#) cntr, const [uint](#) surr, const double val)

- virtual double `getCoeff` (const `uint` cntr, const `uint` surr) const  
*Get the mean/ weighting coefficient for the given center/surround level.*
- virtual double `getMean` (const `uint` cntr, const `uint` surr) const
- virtual double `getSigma` (const `uint` cntr, const `uint` surr) const
- virtual void `learn` (const `Image`< `byte` > &dmap, const double eta, const bool softmax=false, const int inthresh=20, const int outthresh=255)  
*Adjust the weighting coefficients according to the target map.*
- virtual bool `outputAvailable` () const  
*Query whether the channel is ready to give yield valid output.*
- virtual const `Dims` & `getMapDims` () const  
*Return the dimensions of the output image.*
- virtual const `Dims` & `getInputDims` () const  
*Return the dimensions of the most recent input image.*
- virtual const `Image`< `float` > & `getImage` (const `uint` lev) const  
*Get the image from the given level of the internal pyramid.*
- virtual `Image`< `float` > `centerSurround` (const `uint` cntr, const `uint` surr) const  
*Get the center/surround image for the given levels.*
- virtual void `centerSurround` (const `uint` cntr, const `uint` surr, `Image`< `float` > &pos, `Image`< `float` > &neg) const  
*Get the center/surround, split into positive and negative parts.*
- virtual void `fillJet` (const `Point2D` &loc, `Jet`< `float` > &j, `std::vector`< `int` > &index) const  
*Fill the components of the given Jet that relate to this channel.*
- virtual void `fillJetNI` (const `Point2D` &loc, `Jet`< `float` > &j, `std::vector`< `int` > &index) const  
*Fill the components of the given Jet that relate to this channel.*
- virtual void `fillJetSpec` (`JetSpec` &js) const  
*Fill the specification of a Jet with data that relates to this channel.*
- virtual `uint` `numSubmaps` () const  
*Return the number of feature maps represented by this channel.*
- virtual `Image`< `float` > `getSubmap` (const `uint` index) const

*This is just a caching wrapper around [computeSubmap\(\)](#).*

- virtual const char \* [getSubmapName](#) (const uint index) const  
*Get the name/description of the feature map at the given index.*
- virtual void [biasFeatures](#) (const double \*mean, const double \*sigma, int &idx)
- virtual void [getFeatures](#) (const [Point2D](#) &locn, [std::vector](#)< double > &mean) const
- virtual void [findWeight](#) ()  
*Weight of the channel is the maximum weight of its subchannels.*
- virtual bool [reco](#) (const [ParamMap](#) &pmap, double &mismatch)  
*Recognise this location in the channel by checking if it is obj.*
- virtual void [findLikelihood](#) (const [ParamMap](#) &pmap, double &likelihood)  
*what is the probability that the fixation belongs to this object?*
- virtual [Image](#)< float > [getOutput](#) ()  
*This is just a caching wrapper around [combineSubMaps\(\)](#).*
- virtual [Image](#)< float > [combineSubMaps](#) ()  
*Combine all feature maps into a single output map.*
- virtual void [saveResults](#) (const [SharedPtr](#)< [OutputFrameSeries](#) > &ofs)  
*Save our various maps using an [OutputFrameSeries](#).*
- virtual const [ImageSet](#)< float > & [pyramid](#) (const uint index) const  
*Access to the underlying pyramid is provided for completeness.*
- virtual double [pyramidTime](#) (const uint index) const  
*Access to the underlying pyramid times is provided for completeness.*
- virtual const [ImageSet](#)< float > & [clipPyramid](#) () const  
*Access to the underlying clipPyramid.*
- [LevelSpec](#) [getLevelSpec](#) () const  
*get our the [LevelSpec](#)*
- int [getNormType](#) () const  
*get our NormType:*
- void [useSingleOpponence](#) (bool use)  
*should we use single-opponence?*

- virtual void [singleOpponentInput](#) (const [ImageSet](#)< float > &centerPyr, const [ImageSet](#)< float > &surroundPyr, const double t, const [Image](#)< byte > &clipMask)

*Single-opponent SingleChannel requires center and surround pyramids.*

## Protected Member Functions

- virtual void [doInput](#) (const [Image](#)< [PixRGB](#)< float > > \*, const [Image](#)< float > &bwimg, const double t, const [Image](#)< byte > &clipMask)  
*SingleChannel requires only luminance input.*
- virtual void [storePyramid](#) (const [ImageSet](#)< float > &p, const double t)  
*Stores p as the channel's pyramid at time t.*
- virtual void [storePyramid2](#) (const [ImageSet](#)< float > &p, const double t)  
*Stores p as the channel's surround pyramid at time t.*
- virtual void [setClipPyramid](#) (const [Image](#)< byte > &clipMask)  
*make the clipping pyramid from the clip mask*
- virtual void [killCaches](#) ()  
*SingleChannel implements this to clear its cached output.*
- virtual void [setPyramid](#) ([SharedPtr](#)< [PyrBuilder](#)< float > > pyr)  
*Change to use a different pyramid object.*
- virtual [ImageSet](#)< float > & [pyrMut](#) (const [uint](#) index)  
*Mutable access to underlying pyramid.*
- virtual [Image](#)< float > [computeSubmap](#) (const [uint](#) idx) const  
*compute a submap; part of [getSubMap\(\)](#)*
- virtual [Image](#)< float > [getRawCSmap](#) (const [uint](#) idx) const  
*get raw CS map; part of [getSubmap\(\)](#)*
- virtual [Image](#)< float > [postProcessMap](#) (const [Image](#)< float > &smap, const [uint](#) idx) const  
*rescale and post-process raw CS map; part of default [getSubMap\(\)](#)*
- virtual void [start1](#) ()
- virtual void [stop2](#) ()
- [uint](#) [csToIndex](#) ([uint](#) centerlev, [uint](#) surroundlev) const

## Protected Attributes

- [ModelParam](#)< bool > **itsTakeAbs**
- [ModelParam](#)< bool > **itsNormalizeOutput**
- [ModelParam](#)< bool > **itsUseRandom**
- [ModelParam](#)< bool > **itsScaleNoiseToMax**
- [ModelParam](#)< float > **itsLowThresh**
- [ModelParam](#)< bool > **itsRectifyPyramid**
- [ModelParam](#)< bool > **itsUseSplitCS**
- [ModelParam](#)< [LevelSpec](#) > **itsLevelSpec**
- [ModelParam](#)< [MaxNormType](#) > **itsNormType**
- [ModelParam](#)< int > **itsQlen**
- [ModelParam](#)< bool > **itsUseOlderVersion**
- [ModelParam](#)< double > **itsTimeDecay**

*Time decay for the contribution of differences in the pyramid queue.*

- [ModelParam](#)< bool > **itsSaveRawMaps**

*Save our raw pyramid levels?*

- [ModelParam](#)< bool > **itsSaveFeatureMaps**

*Save our center-surround feature maps?*

- [ModelParam](#)< bool > **itsSaveOutputMap**

*Save our output map?*

## Friends

- class [SingleChannelBeo](#)
- class [OrientationChannel](#)
- class [RGBConvolveChannel](#)

### 10.260.1 Detailed Description

SingleChannel represents a single stream of processing.

The processing is implemented by an internal [PyrBuilder](#) object of some kind.

Definition at line 53 of file SingleChannel.H.

## 10.260.2 Constructor & Destructor Documentation

**10.260.2.1** `SingleChannel::SingleChannel (ModelManager & mgr, const std::string & descrName, const std::string & tagName, const VisualFeature vs, SharedPtr< PyrBuilder< float > > pyr)`

Constructor. See [ChannelBase.H](#).

### Parameters:

- mgr* our [ModelManager](#) (see [ModelManager.H](#))
- descrName* descriptive name for human useag
- tagName* name for [ParamMap](#) usage
- vs* The VisualFeature implemented by the channel
- pyr* The type of pyramid that should be used.

Definition at line 57 of file [SingleChannel.C](#).

References [MaxNormType](#), and [VCXNORM\\_FANCY](#).

**10.260.2.2** `SingleChannel::~SingleChannel ()` [virtual]

destructor

Definition at line 91 of file [SingleChannel.C](#).

## 10.260.3 Member Function Documentation

**10.260.3.1** `double SingleChannel::absSumCoeffs () const` [virtual]

Return the sum of the absolute values of the weighting coefficients.

Implements [ChannelBase](#).

Definition at line 262 of file [SingleChannel.C](#).

References [abs\(\)](#), [ModelParam< LevelSpec >::getVal\(\)](#), [LevelSpec::maxIndex\(\)](#), and [uint](#).

**10.260.3.2** `void SingleChannel::biasFeatures (const double * mean, const double * sigma, int & idx)` [virtual]

Bias the features at submap idx with mean, intra-class std. dev and inter-sub-class std. dev

Implements [ChannelBase](#).

Definition at line 762 of file SingleChannel.C.

References ASSERT, ModelParam< LevelSpec >::getVal(), LevelSpec::indexOK(), LevelSpec::indexToCS(), numSubmaps(), setMean(), setSigma(), and uint.

**10.260.3.3** `void SingleChannel::centerSurround (const uint cntr, const uint surr, Image< float > & pos, Image< float > & neg) const` [virtual]

Get the center/surround, split into positive and negative parts.

Definition at line 430 of file SingleChannel.C.

References ASSERT, LevelSpec::csOK(), exp(), ModelParam< double >::getVal(), ModelParam< LevelSpec >::getVal(), itsTimeDecay, outputAvailable(), and uint.

**10.260.3.4** `Image< float > SingleChannel::centerSurround (const uint cntr, const uint surr) const` [virtual]

Get the center/surround image for the given levels.

Reimplemented in [DummyChannel](#).

Definition at line 363 of file SingleChannel.C.

References ASSERT, LevelSpec::csOK(), exp(), ModelParam< double >::getVal(), ModelParam< bool >::getVal(), ModelParam< LevelSpec >::getVal(), itsTimeDecay, outputAvailable(), and uint.

Referenced by getRawCSmap().

**10.260.3.5** `void SingleChannel::clampCoeffs (const double cmin, const double cmax)` [virtual]

Clamp the weighting coefficients to the specified range.

Implements [ChannelBase](#).

Definition at line 249 of file SingleChannel.C.

References CLAMP(), ModelParam< LevelSpec >::getVal(), killCaches(), LevelSpec::maxIndex(), and uint.

**10.260.3.6** `void SingleChannel::cleanCoeffs (const double eps)` [virtual]

Set any coefficients less than eps to zero.

Implements [ChannelBase](#).

Definition at line 236 of file SingleChannel.C.



References `ModelParam< LevelSpec >::getVal()`, `killCaches()`, `LevelSpec::maxIndex()`, and `uint`.

### 10.260.3.7 `const ImageSet< float > & SingleChannel::clipPyramid () const` [virtual]

Access to the underlying `clipPyramid`.

Definition at line 1052 of file `SingleChannel.C`.

### 10.260.3.8 `Image< float > SingleChannel::combineSubMaps ()` [virtual]

Combine all feature maps into a single output map.

Reimplemented in [HueChannel](#).

Definition at line 1064 of file `SingleChannel.C`.

References `ModelManager::debugMode()`, `ModelComponent::descriptiveName()`, `getMapDims()`, `Image< T >::getMinMax()`, `getSubmap()`, `ModelParam< float >::getVal()`, `ModelParam< bool >::getVal()`, `ModelParam< MaxNormType >::getVal()`, `ModelParam< LevelSpec >::getVal()`, `goodness_map()`, `LevelSpec::indexToCS()`, `LevelSpec::maxIndex()`, `maxNormalize()`, `maxNormTypeName()`, `ModelComponent::tagName()`, `uint`, `VCXNORM_LANDMARK`, and `ZEROS`.

Referenced by `getOutput()`, and `FlickerChannel::getOutput()`.

### 10.260.3.9 `ImageSet< float > SingleChannel::computePyramid (const Image< float > & bwimg)` [virtual]

Computes the channel's pyramid but doesn't store it.

Definition at line 958 of file `SingleChannel.C`.

References `doLowThresh()`, `doLowThreshAbs()`, `doRectify()`, `ModelParam< bool >::getVal()`, `ModelParam< float >::getVal()`, `ModelParam< LevelSpec >::getVal()`, and `LevelSpec::maxDepth()`.

Referenced by `doInput()`.

### 10.260.3.10 `Image< float > SingleChannel::computeSubmap (const uint idx)` `const` [protected, virtual]

compute a submap; part of `getSubMap()`

Reimplemented in [IntensityChannelSurprise](#), [ConvolveChannelSurprise](#), [RedGreenChannelSurprise](#), [BlueYellowChannelSurprise](#), [SORedGreenChannelSurprise](#),

[SOGreenRedChannelSurprise](#), [SOBlueYellowChannelSurprise](#), [SOYellowBlueChannelSurprise](#), [RedChannelSurprise](#), [GreenChannelSurprise](#), [BlueChannelSurprise](#), [YellowChannelSurprise](#), [GaborChannelSurprise](#), [FlickerChannelSurprise](#), [ColorBandChannelSurprise](#), [DirectionChannelSurprise](#), [TemplateMatchChannelSurprise](#), [PedestrianChannelSurprise](#), [JunctionChannelSurprise](#), and [DummyChannelSurprise](#).

Definition at line 618 of file `SingleChannel.C`.

References `getRawCSmap()`, `postProcessMap()`, and `uint`.

Referenced by `getSubmap()`.

#### 10.260.3.11 `uint SingleChannel::csToIndex (uint centerlev, uint surroundlev) const` [protected]

If you overload this, make sure you call [SingleChannel::stop1\(\)](#) at the beginning of your overload

Definition at line 1140 of file `SingleChannel.C`.

References `LevelSpec::csToIndex()`, `ModelParam< LevelSpec >::getVal()`, and `uint`.

Referenced by `getCoeff()`, `getMean()`, `getSigma()`, `learn()`, `setCoeff()`, `setMean()`, and `setSigma()`.

#### 10.260.3.12 `void SingleChannel::doInput (const Image< PixRGB< float > > *, const Image< float > * bwing, const double t, const Image< byte > & clipMask)` [protected, virtual]

`SingleChannel` requires only luminance input.

Implements [ChannelBase](#).

Reimplemented in [RGBConvolveChannel](#), [FlickerChannel](#), [DirectionChannel](#), [JunctionChannel](#), [EntropyChannel](#), [VarianceChannel](#), [TcorrChannel](#), [ScorrChannel](#), [InformationChannel](#), [DummyChannel](#), [PN03contrastChannel](#), [DisparityChannel](#), [HueChannel](#), [IntensityChannelBeo](#), [RedGreenChannelBeo](#), [BlueYellowChannelBeo](#), [GaborChannelBeo](#), [FlickerChannelBeo](#), [ColorBandChannelBeo](#), and [DirectionChannelBeo](#).

Definition at line 935 of file `SingleChannel.C`.

References `ASSERT`, `computePyramid()`, `setClipPyramid()`, and `storePyramid()`.

Referenced by `HueChannel::doInput()`, `DirectionChannel::doInput()`, and `FlickerChannel::doInput()`.

### 10.260.3.13 `void SingleChannel::exportOptions (const int whatToExport, const bool recurse = true) [virtual]`

Export some of our [ModelParam](#) parameters as command-line-options.

See the base function in [ModelComponent.H](#) for info.

Reimplemented from [ChannelBase](#).

Reimplemented in [GaborChannel](#), [DirectionChannel](#), [JunctionChannel](#), [TcorrChannel](#), [ScorrChannel](#), [DummyChannel](#), and [DisparityChannel](#).

Definition at line 95 of file [SingleChannel.C](#).

References [ChannelBase::exportOptions\(\)](#), [itsSaveFeatureMaps](#), [itsSaveRawMaps](#), [itsTimeDecay](#), [OPTEXP\\_CORE](#), [OPTEXP\\_SAVE](#), and [ModelManager::requestOption\(\)](#).

Referenced by [DummyChannel::exportOptions\(\)](#), [ScorrChannel::exportOptions\(\)](#), [TcorrChannel::exportOptions\(\)](#), [JunctionChannel::exportOptions\(\)](#), [DirectionChannel::exportOptions\(\)](#), [DisparityChannel::exportOptions\(\)](#), and [GaborChannel::exportOptions\(\)](#).

### 10.260.3.14 `void SingleChannel::fillJet (const Point2D & loc, Jet< float > & j, std::vector< int > & index) const [virtual]`

Fill the components of the given [Jet](#) that relate to this channel.

All relevant components of the [Jet](#) are set to this channel's values at the specified location across all pyramid levels. The channel will store its data starting at the given index in the jet, and will return the number of entries filled. The index vector will be modified as the various [Jet](#) components are explored, but will be restored to its original state upon completion.

Implements [ChannelBase](#).

Definition at line 493 of file [SingleChannel.C](#).

References [LevelSpec::delMax\(\)](#), [Jet< T >::getIndexRange\(\)](#), [Jet< T >::getSpec\(\)](#), [ModelParam< LevelSpec >::getVal\(\)](#), [LevelSpec::levMax\(\)](#), [outputAvailable\(\)](#), [RAW](#), [Jet< T >::setVal\(\)](#), and [ChannelBase::visualFeature\(\)](#).

### 10.260.3.15 `void SingleChannel::fillJetNI (const Point2D & loc, Jet< float > & j, std::vector< int > & index) const [virtual]`

Fill the components of the given [Jet](#) that relate to this channel.

This version does not attempt to use interpolation

Implements [ChannelBase](#).

Definition at line 546 of file SingleChannel.C.

References LevelSpec::delMax(), Jet< T >::getIndexRange(), Jet< T >::getSpec(), ModelParam< LevelSpec >::getVal(), LevelSpec::levMax(), outputAvailable(), RAW, Jet< T >::setVal(), and ChannelBase::visualFeature().

#### 10.260.3.16 void SingleChannel::fillJetSpec (JetSpec & js) const [virtual]

Fill the specification of a Jet with data that relates to this channel.

Implements ChannelBase.

Definition at line 582 of file SingleChannel.C.

References JetSpec::addIndexRange(), LevelSpec::delMax(), ModelParam< LevelSpec >::getVal(), LevelSpec::levMax(), RAW, and ChannelBase::visualFeature().

#### 10.260.3.17 void SingleChannel::findLikelihood (const ParamMap & pmap, double & likelihood) [virtual]

what is the probability that the fixation belongs to this object?

Implements ChannelBase.

Definition at line 884 of file SingleChannel.C.

References LevelSpec::delMax(), LevelSpec::delMin(), getMean(), ParamMap::getSubmap(), ModelParam< LevelSpec >::getVal(), LevelSpec::levMax(), LevelSpec::levMin(), and uint.

#### 10.260.3.18 void SingleChannel::findWeight () [virtual]

Weight of the channel is the maximum weight of its subchannels.

used for biasing to find targets

Implements ChannelBase.

Definition at line 832 of file SingleChannel.C.

References getMean(), getSigma(), ModelParam< LevelSpec >::getVal(), LevelSpec::indexToCS(), numSubmaps(), setCoeff(), ChannelBase::setTotalWeight(), and uint.

#### 10.260.3.19 double SingleChannel::getCoeff (const uint cntr, const uint surr) const [virtual]

Get the mean/ weighting coefficient for the given center/surround level.

Definition at line 302 of file SingleChannel.C.

References [csToIndex\(\)](#), and [uint](#).

Referenced by [writeTo\(\)](#).

#### 10.260.3.20 **const [Image](#)< float > & SingleChannel::getImage (const [uint](#) lev)** **const [virtual]**

Get the image from the given level of the internal pyramid.

Definition at line 356 of file SingleChannel.C.

References [outputAvailable\(\)](#), and [uint](#).

Referenced by [HueChannel::combineSubMaps\(\)](#), and [SoxChannel::getLinearResponse\(\)](#).

#### 10.260.3.21 **const [Dims](#) & SingleChannel::getInputDims () const [virtual]**

Return the dimensions of the most recent input image.

Implements [ChannelBase](#).

Definition at line 349 of file SingleChannel.C.

References [outputAvailable\(\)](#).

Referenced by [TemplateMatchChannel::findBestMatch\(\)](#).

#### 10.260.3.22 **[LevelSpec](#) SingleChannel::getLevelSpec () const**

get our the [LevelSpec](#)

Definition at line 1056 of file SingleChannel.C.

References [ModelParam< LevelSpec >::getVal\(\)](#).

Referenced by [SoxChannel::numScales\(\)](#), and [StereoChannel::start1\(\)](#).

#### 10.260.3.23 **const [Dims](#) & SingleChannel::getMapDims () const [virtual]**

Return the dimensions of the output image.

Implements [ChannelBase](#).

Reimplemented in [DummyChannel](#).

Definition at line 342 of file SingleChannel.C.

References `ModelParam< LevelSpec >::getVal()`, `LevelSpec::mapLevel()`, and `outputAvailable()`.

Referenced by `combineSubMaps()`, `getRawCSmap()`, and `postProcessMap()`.

#### 10.260.3.24 **double SingleChannel::getMean (const uint cntr, const uint surr)** **const [virtual]**

Get the mean feature response for the given center/surround level.

Definition at line 306 of file `SingleChannel.C`.

References `csToIndex()`, and `uint`.

Referenced by `findLikelihood()`, `findWeight()`, `reco()`, and `writeTo()`.

#### 10.260.3.25 **int SingleChannel::getNormType () const**

get our NormType:

Definition at line 1060 of file `SingleChannel.C`.

References `ModelParam< MaxNormType >::getVal()`.

#### 10.260.3.26 **Image< float > SingleChannel::getOutput () [virtual]**

This is just a caching wrapper around `combineSubMaps()`.

Implements `ChannelBase`.

Reimplemented in `FlickerChannel`, `EntropyChannel`, `VarianceChannel`, `TcorrChannel`, `ScorrChannel`, `InformationChannel`, and `PN03contrastChannel`.

Definition at line 1117 of file `SingleChannel.C`.

References `combineSubMaps()`, and `Image< float >::initialized()`.

Referenced by `OrientationChannel::doInteractions()`, `HueChannel::getSubmap()`, and `saveResults()`.

#### 10.260.3.27 **Image< float > SingleChannel::getRawCSmap (const uint idx)** **const [protected, virtual]**

get raw CS map; part of `getSubmap()`

Definition at line 626 of file `SingleChannel.C`.

References `ASSERT`, `binaryReverse()`, `centerSurround()`, `ModelManager::debug-Mode()`, `getMapDims()`, `Image< T >::getMinMax()`, `ModelParam< MaxNormType >::getVal()`, `ModelParam< bool >::getVal()`, `ModelParam< LevelSpec >::getVal()`,

LevelSpec::indexOK(), LevelSpec::indexToCS(), maxNormalize(), outputAvailable(), ModelComponent::tagName(), uint, and ZEROS.

Referenced by computeSubmap().

#### 10.260.3.28 **double SingleChannel::getSigma (const [uint](#) cntr, const [uint](#) surr)** **const** [virtual]

Get the std. dev of the feature response for the given center/surround level.

Definition at line 310 of file SingleChannel.C.

References csToIndex(), and uint.

Referenced by findWeight(), and writeTo().

#### 10.260.3.29 **[Image](#)< [float](#) > SingleChannel::getSubmap (const [uint](#) index) const** **[virtual]**

This is just a caching wrapper around [computeSubmap\(\)](#).

Implements [ChannelBase](#).

Definition at line 608 of file SingleChannel.C.

References computeSubmap(), [Image](#)< [float](#) >::initialized(), and uint.

Referenced by combineSubMaps(), learn(), and saveResults().

#### 10.260.3.30 **const char \* SingleChannel::getSubmapName (const [uint](#) index)** **const** [virtual]

Get the name/description of the feature map at the given index.

This function is intended to be implemented by subclasses so that it recurses into any nested channels, mapping nested submaps into a linear sequence of indices. The output of this function is intended only to be human-readable (e.g. for labeling image displays), and does not have to follow any particular parseable syntax.

Implements [ChannelBase](#).

Reimplemented in [HueChannel](#).

Definition at line 747 of file SingleChannel.C.

References ASSERT, ModelComponent::descriptiveName(), ModelParam< LevelSpec >::getVal(), LevelSpec::indexOK(), LevelSpec::indexToCS(), and uint.

**10.260.3.31** `void SingleChannel::inputPyramid (const ImageSet< float > & pyramid, const double t, const Image< byte > & clipMask = Image< byte >()) [virtual]`

Send an already computed pyramid as input to the channel.

Definition at line 947 of file SingleChannel.C.

References `killCaches()`, `setClipPyramid()`, and `storePyramid()`.

**10.260.3.32** `void SingleChannel::killCaches () [protected, virtual]`

SingleChannel implements this to clear its cached output.

Reimplemented from [ChannelBase](#).

Definition at line 922 of file SingleChannel.C.

References `Image< float >::freeMem()`, `ModelParam< LevelSpec >::getVal()`, `ChannelBase::killCaches()`, `LevelSpec::maxIndex()`, `ModelComponent::started()`, and `uint`.

Referenced by `clampCoeffs()`, `cleanCoeffs()`, `JunctionChannel::doInput()`, `inputPyramid()`, `learn()`, `normalizeCoeffs()`, `SingleChannelBeo::sendInput()`, `setCoeff()`, `setMean()`, `setSigma()`, and `singleOpponentInput()`.

**10.260.3.33** `void SingleChannel::learn (const Image< byte > & dmap, const double eta, const bool softmax = false, const int inthresh = 20, const int outthresh = 255) [virtual]`

Adjust the weighting coefficients according to the target map.

Implements [ChannelBase](#).

Definition at line 314 of file SingleChannel.C.

References `csToIndex()`, `LevelSpec::delMax()`, `LevelSpec::delMin()`, `Image< T >::getDims()`, `getSubmap()`, `ModelParam< LevelSpec >::getVal()`, `killCaches()`, `learningCoeff()`, `LevelSpec::levMax()`, `LevelSpec::levMin()`, `rescale()`, `ModelComponent::tagName()`, and `uint`.

**10.260.3.34** `void SingleChannel::normalizeCoeffs (const double div) [virtual]`

Normalize the weighting coefficients by dividing by div.

Implements [ChannelBase](#).

Definition at line 271 of file SingleChannel.C.



References `ModelParam< LevelSpec >::getVal()`, `killCaches()`, `LevelSpec::maxIndex()`, and `uint`.

#### 10.260.3.35 `uint SingleChannel::numChans () const` [virtual]

number of subchannels for [ComplexChannel](#), 0 for `SingleChannel`

Implements [ChannelBase](#).

Definition at line 232 of file `SingleChannel.C`.

References `uint`.

#### 10.260.3.36 `uint SingleChannel::numSubmaps () const` [virtual]

Return the number of feature maps represented by this channel.

Implements [ChannelBase](#).

Reimplemented in [HueChannel](#).

Definition at line 604 of file `SingleChannel.C`.

References `ModelParam< LevelSpec >::getVal()`, `LevelSpec::maxIndex()`, and `uint`.

Referenced by `biasFeatures()`, `findWeight()`, `saveResults()`, and `SingleChannelBeo::waitForOutput()`.

#### 10.260.3.37 `bool SingleChannel::outputAvailable () const` [virtual]

Query whether the channel is ready to give yield valid output.

In some cases (e.g. with motion channels), it may take several input frames before the channel is fully initialized.

Implements [ChannelBase](#).

Reimplemented in [EntropyChannel](#), [VarianceChannel](#), [TcorrChannel](#), [ScorrChannel](#), [InformationChannel](#), [DummyChannel](#), [PN03contrastChannel](#), [IntensityChannelBeo](#), [RedGreenChannelBeo](#), [BlueYellowChannelBeo](#), [GaborChannelBeo](#), [FlickerChannelBeo](#), [ColorBandChannelBeo](#), and [DirectionChannelBeo](#).

Definition at line 338 of file `SingleChannel.C`.

Referenced by `centerSurround()`, `fillJet()`, `fillJetNI()`, `getImage()`, `getInputDims()`, `getMapDims()`, `FlickerChannel::getOutput()`, `getRawCSmap()`, `DirectionChannelBeo::outputAvailable()`, `ColorBandChannelBeo::outputAvailable()`, `FlickerChannelBeo::outputAvailable()`, `GaborChannelBeo::outputAvailable()`, `BlueYellowChannelBeo::outputAvailable()`, `RedGreenChannelBeo::outputAvailable()`, `IntensityChannelBeo::outputAvailable()`, `pyramid()`, and `saveResults()`.

### 10.260.3.38 **Image**< float > SingleChannel::postProcessMap (const **Image**< float > & *smap*, const **uint** *idx*) const [protected, virtual]

rescale and post-process raw CS map; part of default getSubMap()

Definition at line 688 of file SingleChannel.C.

References `Image< T >::addBGnoise()`, `ModelManager::debugMode()`, `downSize()`, `SharedPtr< ChannelListener >::get()`, `getMapDims()`, `Image< T >::getMinMax()`, `ModelParam< MaxNormType >::getVal()`, `ModelParam< bool >::getVal()`, `ModelParam< LevelSpec >::getVal()`, `Image< T >::getWidth()`, `LevelSpec::indexToCS()`, `maxNormalize()`, `MAXNORMMAX`, `MAXNORMMIN`, `maxNormType`, `Name()`, `rescale()`, `ModelComponent::tagName()`, `uint`, and `Dims::w()`.

Referenced by `computeSubmap()`.

### 10.260.3.39 **const ImageSet**< float > & SingleChannel::pyramid (const **uint** *index*) const [virtual]

Access to the underlying pyramid is provided for completeness.

However, it probably shouldn't need to be used except for testing, debugging, etc. Instead, the Channel interface should be used or extended to fill the specific need.

#### Parameters:

***index*** The index in the queue (0 = newest, qlen-1 = oldest).

Definition at line 1036 of file SingleChannel.C.

References `ASSERT`, `outputAvailable()`, and `uint`.

Referenced by `ShapeEstimator::getPyramid()`.

### 10.260.3.40 **double** SingleChannel::pyramidTime (const **uint** *index*) const [virtual]

Access to the underlying pyramid times is provided for completeness.

However, it probably shouldn't need to be used except for testing, debugging, etc. Instead, the Channel interface should be used or extended to fill the specific need.

#### Parameters:

***index*** The index in the queue (0 = newest, qlen-1 = oldest).

Definition at line 1044 of file SingleChannel.C.

References `ASSERT`, and `uint`.

**10.260.3.41** [ImageSet](#)< float > & SingleChannel::pyrMut (const [uint](#) *index*)  
[protected, virtual]

Mutable access to underlying pyramid.

Definition at line 1133 of file SingleChannel.C.

References [ASSERT](#), and [uint](#).

Referenced by [DirectionChannel::doInput\(\)](#).

**10.260.3.42** void SingleChannel::readFrom (const [ParamMap](#) & *pmap*)  
[virtual]

Read params from the [ParamMap](#).

Reimplemented from [ChannelBase](#).

Reimplemented in [GaborChannel](#).

Definition at line 166 of file SingleChannel.C.

References [LevelSpec::delMax\(\)](#), [LevelSpec::delMin\(\)](#), [ParamMap::getDoubleParam\(\)](#), [ParamMap::getSubmap\(\)](#), [ModelParam< LevelSpec >::getVal\(\)](#), [ParamMap::hasParam\(\)](#), [ParamMap::isLeaf\(\)](#), [LevelSpec::levMax\(\)](#), [LevelSpec::levMin\(\)](#), [ChannelBase::readFrom\(\)](#), [setCoeff\(\)](#), [setMean\(\)](#), [setSigma\(\)](#), and [uint](#).

Referenced by [GaborChannel::readFrom\(\)](#).

**10.260.3.43** bool SingleChannel::reco (const [ParamMap](#) & *pmap*, double & *mismatch*) [virtual]

Recognise this location in the channel by checking if it is obj.

Implements [ChannelBase](#).

Definition at line 859 of file SingleChannel.C.

References [LevelSpec::delMax\(\)](#), [LevelSpec::delMin\(\)](#), [getMean\(\)](#), [ParamMap::getSubmap\(\)](#), [ModelParam< LevelSpec >::getVal\(\)](#), [LevelSpec::levMax\(\)](#), [LevelSpec::levMin\(\)](#), and [uint](#).

**10.260.3.44** void SingleChannel::reset (const bool *recurse* = true) [virtual]

Reset SingleChannel.

See the base function in [ModelComponent.H](#) for info.

Reimplemented from [ModelComponent](#).

Reimplemented in [FlickerChannel](#).

Definition at line 146 of file SingleChannel.C.

References `ImageSet< float >::clear()`, `Image< float >::freeMem()`, `ModelParam< LevelSpec >::getVal()`, `LevelSpec::maxIndex()`, `ModelComponent::reset()`, `SharedPtr< PyrBuilder< float > >::reset()`, and `uint`.

Referenced by `FlickerChannel::reset()`.

#### 10.260.3.45 `void SingleChannel::saveResults (const SharedPtr< OutputFrameSeries > & ofs) [virtual]`

Save our various maps using an `OutputFrameSeries`.

Depending on our `ModelParam` settings, we can save raw pyramid levels (with name prefix "SR<tagname>-<level>-"), center-surround feature maps ("SF<tagname>-<ctr>-<surr>-"), and output map ("SO<tagname>-")

Reimplemented from `ChannelBase`.

Definition at line 1145 of file SingleChannel.C.

References `getOutput()`, `getSubmap()`, `ModelParam< LevelSpec >::getVal()`, `ModelParam< bool >::getVal()`, `LevelSpec::indexToCS()`, `itsSaveFeatureMaps`, `itsSaveRawMaps`, `numSubmaps()`, `outputAvailable()`, `ImageSet< T >::size()`, `ModelComponent::tagName()`, and `uint`.

#### 10.260.3.46 `void SingleChannel::setClipPyramid (const Image< byte > & clipMask) [protected, virtual]`

make the clipping pyramid from the clip mask

Definition at line 980 of file SingleChannel.C.

References `buildPyrGaussian()`, `ImageSet< float >::clear()`, `doLowThresh()`, `ModelParam< LevelSpec >::getVal()`, `Image< T >::initialized()`, and `LevelSpec::maxDepth()`.

Referenced by `doInput()`, `JunctionChannel::doInput()`, `RGBConvolveChannel::doInput()`, `inputPyramid()`, and `singleOpponentInput()`.

#### 10.260.3.47 `void SingleChannel::setCoeff (const uint cntr, const uint surr, const double val) [virtual]`

Set the weighting coefficient for the given center/surround level.

Definition at line 279 of file SingleChannel.C.

References `csToIndex()`, `killCaches()`, and `uint`.

Referenced by `findWeight()`, and `readFrom()`.

**10.260.3.48** `void SingleChannel::setListener (SharedPtr< ChannelListener > listener) [virtual]`

Set a [ChannelListener](#) for this channel.

The [ChannelListener](#) which will get called once for each center/surround pairing.

Implements [ChannelBase](#).

Definition at line 228 of file SingleChannel.C.

**10.260.3.49** `void SingleChannel::setMean (const uint cntr, const uint surr, const double val) [virtual]`

Set the mean feature response for the given center/surround level.

Definition at line 286 of file SingleChannel.C.

References [csToIndex\(\)](#), [killCaches\(\)](#), and [uint](#).

Referenced by [biasFeatures\(\)](#), and [readFrom\(\)](#).

**10.260.3.50** `void SingleChannel::setPyramid (SharedPtr< PyrBuilder< float > > pyr) [protected, virtual]`

Change to use a different pyramid object.

Definition at line 1125 of file SingleChannel.C.

References [ASSERT](#).

Referenced by [ConvolveChannel::setFilter\(\)](#), and [RGBConvolveChannel::setFilters\(\)](#).

**10.260.3.51** `void SingleChannel::setSigma (const uint cntr, const uint surr, const double val) [virtual]`

Set the std. dev of feature response at the given center/surround level.

Definition at line 294 of file SingleChannel.C.

References [csToIndex\(\)](#), [killCaches\(\)](#), and [uint](#).

Referenced by [biasFeatures\(\)](#), and [readFrom\(\)](#).

**10.260.3.52** `void SingleChannel::singleOpponentInput (const ImageSet< float > & centerPyr, const ImageSet< float > & surroundPyr, const double t, const Image< byte > & clipMask) [virtual]`

Single-opponent SingleChannel requires center and surround pyramids.

Definition at line 1177 of file SingleChannel.C.

References ASSERT, killCaches(), setClipPyramid(), storePyramid(), and storePyramid2().

Referenced by SOColorChannel::doInput().

### 10.260.3.53 void SingleChannel::start1 () [protected, virtual]

If you overload this, make sure you call [SingleChannel::start1\(\)](#) at the beginning of your overload

Reimplemented from [ModelComponent](#).

Reimplemented in [GaborChannel](#), [TemplateMatchChannel](#), [PedestrianChannel](#), [DisparityChannel](#), [IntensityChannelBeo](#), [RedGreenChannelBeo](#), [BlueYellowChannelBeo](#), [GaborChannelBeo](#), [FlickerChannelBeo](#), [ColorBandChannelBeo](#), and [DirectionChannelBeo](#).

Definition at line 118 of file SingleChannel.C.

References ModelParam< MaxNormType >::getVal(), ModelParam< bool >::getVal(), ModelParam< LevelSpec >::getVal(), LevelSpec::maxIndex(), ChannelBase::setOutputRange(), and VCXNORM\_MAXNORM.

Referenced by DirectionChannelBeo::start1(), ColorBandChannelBeo::start1(), FlickerChannelBeo::start1(), BlueYellowChannelBeo::start1(), RedGreenChannelBeo::start1(), IntensityChannelBeo::start1(), TemplateMatchChannel::start1(), DisparityChannel::start1(), and GaborChannel::start1().

### 10.260.3.54 void SingleChannel::stop2 () [protected, virtual]

If you overload this, make sure you call [SingleChannel::start1\(\)](#) at the beginning of your overload

Reimplemented from [ModelComponent](#).

Definition at line 137 of file SingleChannel.C.

### 10.260.3.55 void SingleChannel::storePyramid (const [ImageSet](#)< float > & p, const double t) [protected, virtual]

Stores p as the channel's pyramid at time t.

Definition at line 994 of file SingleChannel.C.

References ASSERT, ModelParam< int >::getVal(), and isDyadic().

Referenced by doInput(), JunctionChannel::doInput(), DisparityChannel::doInput(), RGBConvolveChannel::doInput(), inputPyramid(), and singleOpponentInput().

**10.260.3.56 void SingleChannel::storePyramid2 (const [ImageSet](#)< float > & p, const double t) [protected, virtual]**

Stores p as the channel's surround pyramid at time t.

This will throw a fatal error if the channel has not been declared as single-opponent through a call to [useSingleOpponence\(\)](#)

Definition at line 1014 of file SingleChannel.C.

References [ASSERT](#), [ModelParam](#)< int >::getVal(), and [isDyadic\(\)](#).

Referenced by [singleOpponentInput\(\)](#).

**10.260.3.57 void SingleChannel::useSingleOpponence (bool use)**

should we use single-opponence?

Definition at line 1171 of file SingleChannel.C.

Referenced by [SOBlueYellowChannel::SOBlueYellowChannel\(\)](#), [SOGreenRedChannel::SOGreenRedChannel\(\)](#), [SORedGreenChannel::SORedGreenChannel\(\)](#), and [SOYellowBlueChannel::SOYellowBlueChannel\(\)](#).

**10.260.3.58 void SingleChannel::writeTo ([ParamMap](#) & pmap) const [virtual]**

Write params to the [ParamMap](#).

Reimplemented from [ChannelBase](#).

Reimplemented in [GaborChannel](#).

Definition at line 205 of file SingleChannel.C.

References [LevelSpec::delMax\(\)](#), [LevelSpec::delMin\(\)](#), [getCoeff\(\)](#), [getMean\(\)](#), [getSigma\(\)](#), [ModelParam](#)< [LevelSpec](#) >::getVal(), [LevelSpec::levMax\(\)](#), [LevelSpec::levMin\(\)](#), [ParamMap::putSubmap\(\)](#), [uint](#), and [ChannelBase::writeTo\(\)](#).

Referenced by [SingleChannelBeo::sendInput\(\)](#), and [GaborChannel::writeTo\(\)](#).

**10.260.4 Member Data Documentation****10.260.4.1 [ModelParam](#)<bool> [SingleChannel::itsSaveFeatureMaps](#) [protected]**

Save our center-surround feature maps?

Definition at line 279 of file SingleChannel.H.

Referenced by `exportOptions()`, and `saveResults()`.

**10.260.4.2**   [\*\*ModelParam<bool> SingleChannel::itsSaveOutputMap\*\*](#)  
[protected]

Save our output map?

Definition at line 282 of file `SingleChannel.H`.

**10.260.4.3**   [\*\*ModelParam<bool> SingleChannel::itsSaveRawMaps\*\*](#)  
[protected]

Save our raw pyramid levels?

Definition at line 276 of file `SingleChannel.H`.

Referenced by `exportOptions()`, and `saveResults()`.

**10.260.4.4**   [\*\*ModelParam<double> SingleChannel::itsTimeDecay\*\*](#)  
[protected]

Time decay for the contribution of differences in the pyramid queue.

This decay will yield a factor  $fac = \exp((t_2 - t) * decay)$  applied to the difference image between  $t$  (current time, in seconds) and  $t_2$  (time of previous image, in seconds):

Definition at line 273 of file `SingleChannel.H`.

Referenced by `centerSurround()`, and `exportOptions()`.

The documentation for this class was generated from the following files:

- [SingleChannel.H](#)
- [SingleChannel.C](#)



## 10.261 SingleChannelBeo Class Reference

Wrapper to run a [SingleChannel](#) on a distant CPU.

```
#include <SingleChannelBeo.H>
```

Inheritance diagram for SingleChannelBeo: Collaboration diagram for SingleChannelBeo:



### Public Member Functions

- [SingleChannelBeo](#) ([SingleChannel](#) \*chan, const [SharedPtr< Beowulf >](#) &beow)  
*Default constructor.*
- void [requestBeowulfNode](#) ()  
*Get a [Beowulf](#) node for our [SingleChannelBeoServer](#).*
- virtual [~SingleChannelBeo](#) ()  
*Virtual destructor ensures proper destruction of derived classes.*
- void [sendInput](#) (const [Image< float >](#) \*bwimg, const double t, const [Image< byte >](#) &clipMask)  
*Send the input image off to the processing node.*
- void [waitForOutput](#) ()  
*Have we received the results from our processing node yet?*

### 10.261.1 Detailed Description

Wrapper to run a [SingleChannel](#) on a distant CPU.

This class is to be used as an add-on to [SingleChannel](#) or derivatives, using multiple inheritance to create classes that derive both from [SingleChannel](#) and [SingleChannelBeo](#). It uses a [Beowulf](#) to delegate the actual feature map computations to a distant CPU. The distant CPU should run a [SingleChannelBeoServer](#), like in the the beochannel-server

program; see [SingleChannelBeoServer.H](#) and `beochannel-server.C` for details. When we get an input image, we will send it off to the processing node specified at construction (and possibly modified later using standard [ModelParam](#) access functions), using the [Beowulf](#) object given at construction. Before sending off our first input, we will contact our processing node and send it all of our associated [SingleChannel](#)'s [ModelParam](#) values in a configuration message; this will allow it to build the actual [SingleChannel](#) which we are associated with.

The system can work in two modes: in full emulation mode, the output map, channel pyramid, submaps and clip pyramid are all returned by the processor node. This requires a lot of network bandwidth but should ensure results that are exactly identical to when using the normal channels. In quick mode, only the output submap is returned. This will work with standard processing that only deals with the output, but things like saving the submaps or biasing the visual cortex will break; so use the fast mode at your own risk ;-). Selection of which mode is used is made by the [SingleChannelBeoServer](#); here we just store whatever the server sends us, be it full or partial results.

PROGRAMMER NOTE: Use this to build channels that derive both from [SingleChannel](#) and [SingleChannelBeo](#). There are only three [SingleChannel](#) methods which you need to overload in your multiply-derived class: First, your overload of [SingleChannel::doInput\(\)](#) should call our [sendInput\(\)](#) instead and should not call [SingleChannel::doInput\(\)](#), since this is the function that in [SingleChannel](#) does the processing which here we want to off-load to a distant CPU. Second, you should overload [SingleChannel::outputAvailable\(\)](#) such as to call our [waitForOutput\(\)](#) first, then call [SingleChannel::outputAvailable\(\)](#). Third, be sure to call our [requestBeowulfNode\(\)](#) some time after your [Beowulf](#) is start()'ed; typically you can do that in an overload of [SingleChannel::start1\(\)](#), calling the [SingleChannel](#)'s `start1()` and our [requestBeowulfNode\(\)](#). See the examples in [ChannelsBeo.H](#) for how this is done.

Definition at line 94 of file [SingleChannelBeo.H](#).

## 10.261.2 Constructor & Destructor Documentation

### 10.261.2.1 [SingleChannelBeo::SingleChannelBeo](#) ([SingleChannel](#) \* *chan*, const [SharedPtr](#)< [Beowulf](#) > & *beow*)

Default constructor.

You will need to call `setBeowulf` before `start()` to properly configure the object.

#### Parameters:

- chan* pointer to the actual [SingleChannel](#); typically, you should pass "this" here when you construct a class that inherits both from [SingleChannel](#) and [SingleChannelBeo](#).
- beo* the [Beowulf](#) to use for data transfers

Definition at line 46 of file [SingleChannelBeo.C](#).

### 10.261.2.2 SingleChannelBeo::~~SingleChannelBeo () [virtual]

Virtual destructor ensures proper destruction of derived classes.

Definition at line 65 of file SingleChannelBeo.C.

## 10.261.3 Member Function Documentation

### 10.261.3.1 void SingleChannelBeo::requestBeowulfNode ()

Get a [Beowulf](#) node for our [SingleChannelBeoServer](#).

During that step, we will query our [Beowulf](#) for the next available node number and will use that as our [SingleChannelBeoServer](#) node. If the [Beowulf](#) has no more available node, we will throw a fatal error. This should be called once the [Beowulf](#) has been start()'ed.

Definition at line 52 of file SingleChannelBeo.C.

References [ModelComponent::descriptiveName\(\)](#), and [ModelComponent::tagName\(\)](#).

Referenced by [DirectionChannelBeo::start1\(\)](#), [ColorBandChannelBeo::start1\(\)](#), [FlickerChannelBeo::start1\(\)](#), [GaborChannelBeo::start1\(\)](#), [BlueYellowChannelBeo::start1\(\)](#), [RedGreenChannelBeo::start1\(\)](#), and [IntensityChannelBeo::start1\(\)](#).

### 10.261.3.2 void SingleChannelBeo::sendInput (const [Image](#)< float > \* *bwimg*, const double *t*, const [Image](#)< byte > & *clipMask*)

Send the input image off to the processing node.

You should call this from inside your overload of [SingleChannel::doInput\(\)](#)

Definition at line 69 of file SingleChannelBeo.C.

References [TCPmessage::addDouble\(\)](#), [TCPmessage::addImage\(\)](#), [TCPmessage::addString\(\)](#), [ModelComponent::descriptiveName\(\)](#), [ParamMap::format\(\)](#), [SingleChannel::killCaches\(\)](#), [ModelComponent::tagName\(\)](#), [ChannelBase::visualFeature\(\)](#), [ModelComponent::writeParamsTo\(\)](#), and [SingleChannel::writeTo\(\)](#).

Referenced by [DirectionChannelBeo::doInput\(\)](#), [ColorBandChannelBeo::doInput\(\)](#), [FlickerChannelBeo::doInput\(\)](#), [GaborChannelBeo::doInput\(\)](#), [BlueYellowChannelBeo::doInput\(\)](#), [RedGreenChannelBeo::doInput\(\)](#), and [IntensityChannelBeo::doInput\(\)](#).

### 10.261.3.3 void SingleChannelBeo::waitForOutput ()

Have we received the results from our processing node yet?

This call will block until the results are received back from the processing node. You should call this from within your overload of [SingleChannel::outputAvailable\(\)](#). This is how we implement the remote processing with minimum differences between [SingleChannel](#) and objects that inherit from both [SingleChannel](#) and `SingleChannelBeo`; indeed, each time a result is requested from [SingleChannel](#), it will (and should in future methods!) check first that some output is available. So, by intercepting this check we can wait until the results have been received from the processing node.

Definition at line 117 of file `SingleChannelBeo.C`.

References `ASSERT`, `TCPmessage::getElement()`, `ModelParam< int >::getVal()`, `Image< float >::initialized()`, `SingleChannel::itsClipPyr`, `SingleChannel::itsOutputCache`, `SingleChannel::itsPq`, `SingleChannel::itsQlen`, `SingleChannel::itsSubmapCache`, `SingleChannel::numSubmaps()`, and `uint`.

Referenced by `FlickerChannelBeo::outputAvailable()`.

The documentation for this class was generated from the following files:

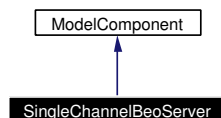
- [SingleChannelBeo.H](#)
- [SingleChannelBeo.C](#)

## 10.262 SingleChannelBeoServer Class Reference

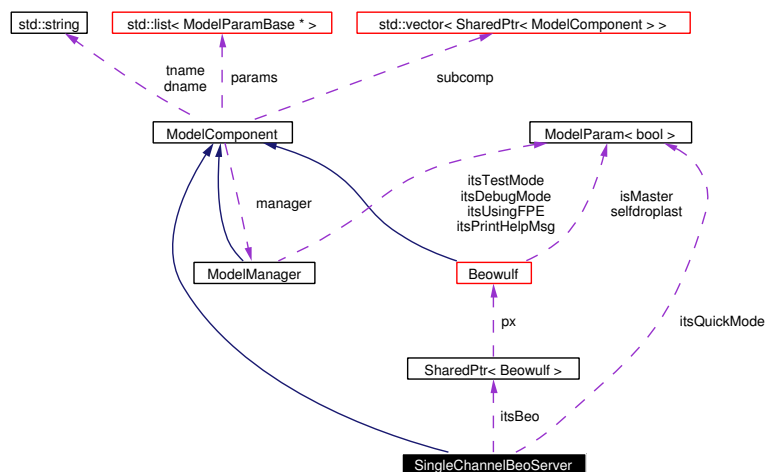
Server to work with [SingleChannelBeo](#).

```
#include <SingleChannelBeoServer.H>
```

Inheritance diagram for SingleChannelBeoServer:



Collaboration diagram for SingleChannelBeoServer:



### Public Member Functions

- `SingleChannelBeoServer` (`ModelManager` &mgr, const `std::string` &descr-Name="SingleChannelBeo Server", const `std::string` &tagName="Single-ChannelBeoServer")

*Constructor.*

- virtual `~SingleChannelBeoServer` ()

*Destructor.*

- virtual void `exportOptions` (const int whatToExport, const bool recurse=true)

Export some of our [ModelParam](#) parameters as command-line-options.

- void [check](#) ()

*Call this often for us to do our work.*

## Protected Attributes

- [ModelParam](#)< bool > [itsQuickMode](#)

*only send off output map, not other maps*

### 10.262.1 Detailed Description

Server to work with [SingleChannelBeo](#).

This class receives images from [SingleChannelBeo](#), process them, and return the pyramids and submaps back. One server may serve several [SingleChannelBeo](#) clients; messages will be processed according to the [Beowulf](#) node number of the client. The first thing the client should do is to sent a configuration message, according to which we will build the channel here. Then, the client may send input images, which we will process, and we will return the resulting pyramid and maps.

The system can work in two modes: in full emulation mode, the output map, channel pyramid, submaps and clip pyramid are all returned by the processor node. This requires a lot of network bandwidth but should ensure results that are exactly identical to when using the normal channels. In quick mode, only the output submap is returned. This will work with standard processing that only deals with the output, but things like saving the submaps or biasing the visual cortex will break; so use the fast mode at your own risk ;-). Selection of which mode is used is done via [opur itsQuickMode ModelParam](#).

Definition at line 68 of file [SingleChannelBeoServer.H](#).

### 10.262.2 Constructor & Destructor Documentation

#### 10.262.2.1 [SingleChannelBeoServer::SingleChannelBeoServer](#) ([ModelManager](#) & *mgr*, const [std::string](#) & *descrName* = "SingleChannelBeoServer", const [std::string](#) & *tagName* = "SingleChannelBeoServer")

Constructor.

Definition at line 50 of file [SingleChannelBeoServer.C](#).

References [ModelComponent::addSubComponent\(\)](#).

### 10.262.2.2 SingleChannelBeoServer::~~SingleChannelBeoServer () [virtual]

Destructor.

Definition at line 62 of file SingleChannelBeoServer.C.

## 10.262.3 Member Function Documentation

### 10.262.3.1 void SingleChannelBeoServer::check ()

Call this often for us to do our work.

Definition at line 79 of file SingleChannelBeoServer.C.

References TCPmessage::addDouble(), TCPmessage::addImage(), TCPmessage::addImageSet(), ModelComponent::addSubComponent(), ASSERT, BY, COLBAND, FLICKER, Gaussian9, Timer::get(), SharedPtr< T >::get(), TCPmessage::getElement(), ModelParam< bool >::getVal(), Dims::h(), INTENS, itsQuickMode, ParamMap::load(), ORI, ModelComponent::removeAllSubComponents(), ModelComponent::removeSubComponent(), TCPmessage::reset(), SharedPtr< T >::reset(), RG, uint, VisualFeature, and Dims::w().

### 10.262.3.2 void SingleChannelBeoServer::exportOptions (const int *whatToExport*, const bool *recurse* = true) [virtual]

Export some of our [ModelParam](#) parameters as command-line-options.

See the base function in [ModelComponent.H](#) for info.

Reimplemented from [ModelComponent](#).

Definition at line 66 of file SingleChannelBeoServer.C.

References ModelComponent::exportOptions(), itsQuickMode, OPTEXP\_CORE, and ModelManager::requestOption().

## 10.262.4 Member Data Documentation

### 10.262.4.1 [ModelParam<bool>](#) [SingleChannelBeoServer::itsQuickMode](#) [protected]

only send off output map, not other maps

Definition at line 90 of file SingleChannelBeoServer.H.

Referenced by check(), and exportOptions().

The documentation for this class was generated from the following files:

- [SingleChannelBeoServer.H](#)
- [SingleChannelBeoServer.C](#)

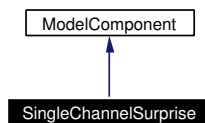


## 10.263 SingleChannelSurprise Class Reference

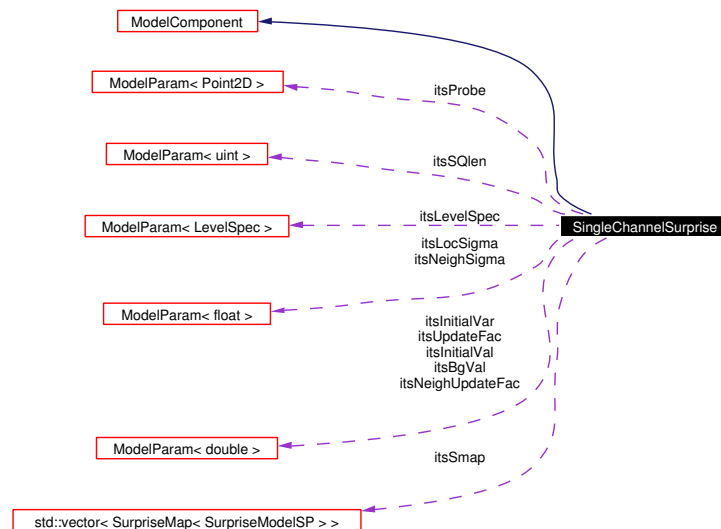
SingleChannelSurprise is a Surprise extension to [SingleChannel](#).

```
#include <SingleChannelSurprise.H>
```

Inheritance diagram for SingleChannelSurprise:



Collaboration diagram for SingleChannelSurprise:



### Public Member Functions

- [SingleChannelSurprise](#) ([ModelManager](#) &mgr, const `std::string` &descrName, const `std::string` &tagName)

*Constructor.*

- virtual [~SingleChannelSurprise](#) ()

*destructor*

- virtual void [exportOptions](#) (const int whatToExport, const bool recurse=true)  
*Export some of our [ModelParam](#) parameters as command-line-options.*
- virtual [Image](#)< float > [getSurpriseMap](#) (const [uint](#) index, const [Image](#)< float > &submap)  
*get a surprise map*

## Protected Member Functions

- virtual void [start1](#) ()  
*This is called from within [start\(\)](#) before the subcomponents start.*

## Protected Attributes

- [ModelParam](#)< [uint](#) > [itsSQLen](#)  
*number of surprise maps*
- [ModelParam](#)< double > [itsUpdateFac](#)  
*update factor*
- [ModelParam](#)< double > [itsNeighUpdateFac](#)  
*update factor for neighbors*
- [ModelParam](#)< double > [itsInitialVal](#)  
*Initial values.*
- [ModelParam](#)< double > [itsInitialVar](#)  
*Initial variance.*
- [ModelParam](#)< double > [itsBgVal](#)  
*Background values.*
- [ModelParam](#)< float > [itsNeighSigma](#)  
*Neighborhood sigma.*
- [ModelParam](#)< float > [itsLocSigma](#)  
*Local neighborhood sigma.*
- [ModelParam](#)< [LevelSpec](#) > [itsLevelSpec](#)  
*our levelspec*

- [ModelParam](#)< [Point2D](#) > [itsProbe](#)

*location of a virtual electrode*

### 10.263.1 Detailed Description

SingleChannelSurprise is a Surprise extension to [SingleChannel](#).

This class only provides the computation of surprise. Typically you would first build a [SingleChannel](#) (or derivative), then build a SingleChannelSurprise, then adopt the SingleChannelSurprise into the [SingleChannel](#) as a subcomponent, then register your [SingleChannel](#) with your [VisualCortex](#), [ModelManager](#) or whoever (but do not register the adopted SingleChannelSurprise object).

Definition at line 53 of file SingleChannelSurprise.H.

### 10.263.2 Constructor & Destructor Documentation

#### 10.263.2.1 SingleChannelSurprise::SingleChannelSurprise ([ModelManager](#) & mgr, const [std::string](#) & descrName, const [std::string](#) & tagName)

Constructor.

Definition at line 42 of file SingleChannelSurprise.C.

#### 10.263.2.2 SingleChannelSurprise::~SingleChannelSurprise () [virtual]

destructor

Definition at line 60 of file SingleChannelSurprise.C.

### 10.263.3 Member Function Documentation

#### 10.263.3.1 void SingleChannelSurprise::exportOptions (const int whatToExport, const bool recurse = true) [virtual]

Export some of our [ModelParam](#) parameters as command-line-options.

See the base function in [ModelComponent.H](#) for info.

Reimplemented from [ModelComponent](#).

Definition at line 64 of file SingleChannelSurprise.C.

References `ModelComponent::exportOptions()`, `itsNeighUpdateFac`, `itsProbe`, `itsSQLen`, `itsUpdateFac`, `OPTEXP_CORE`, and `ModelManager::requestOption()`.

#### 10.263.3.2 `Image< float > SingleChannelSurprise::getSurpriseMap (const uint index, const Image< float > & submap) [virtual]`

get a surprise map

Pass a submap index and a submap, and you'll get back the associated surprise

Definition at line 96 of file `SingleChannelSurprise.C`.

References `Image< T >::clear()`, `Image< T >::getDims()`, `ModelParam< double >::getVal()`, `itsBgVal`, `itsInitialVal`, `itsUpdateFac`, `NO_INIT`, and `uint`.

#### 10.263.3.3 `void SingleChannelSurprise::start1 () [protected, virtual]`

This is called from within `start()` before the subcomponents start.

Reimplemented from `ModelComponent`.

Definition at line 81 of file `SingleChannelSurprise.C`.

References `ModelParam< Point2D >::getVal()`, `ModelParam< float >::getVal()`, `ModelParam< double >::getVal()`, `ModelParam< uint >::getVal()`, `ModelParam< LevelSpec >::getVal()`, `SurpriseMap< T >::init()`, `itsBgVal`, `itsInitialVal`, `itsInitialVal`, `itsLocSigma`, `itsNeighSigma`, `itsNeighUpdateFac`, `itsProbe`, `itsSQLen`, `itsUpdateFac`, and `LevelSpec::maxIndex()`.

### 10.263.4 Member Data Documentation

#### 10.263.4.1 `ModelParam<double> SingleChannelSurprise::itsBgVal [protected]`

Background values.

Definition at line 80 of file `SingleChannelSurprise.H`.

Referenced by `getSurpriseMap()`, and `start1()`.

#### 10.263.4.2 `ModelParam<double> SingleChannelSurprise::itsInitialVal [protected]`

Initial values.

Definition at line 78 of file `SingleChannelSurprise.H`.

Referenced by `start1()`.

**10.263.4.3** [ModelParam<double> SingleChannelSurprise::itsInitialVar](#)  
[protected]

Initial variance.

Definition at line 79 of file SingleChannelSurprise.H.

Referenced by getSurpriseMap(), and start1().

**10.263.4.4** [ModelParam<LevelSpec> SingleChannelSurprise::itsLevelSpec](#)  
[protected]

our levelspec

Definition at line 83 of file SingleChannelSurprise.H.

**10.263.4.5** [ModelParam<float> SingleChannelSurprise::itsLocSigma](#)  
[protected]

Local neighborhood sigma.

Definition at line 82 of file SingleChannelSurprise.H.

Referenced by start1().

**10.263.4.6** [ModelParam<float> SingleChannelSurprise::itsNeighSigma](#)  
[protected]

Neighborhood sigma.

Definition at line 81 of file SingleChannelSurprise.H.

Referenced by start1().

**10.263.4.7** [ModelParam<double> SingleChannelSurprise::itsNeighUpdateFac](#)  
[protected]

update factor for neighbors

Definition at line 77 of file SingleChannelSurprise.H.

Referenced by exportOptions(), and start1().

**10.263.4.8** [ModelParam<Point2D> SingleChannelSurprise::itsProbe](#)  
[protected]

location of a virtusl electrode

Definition at line 84 of file SingleChannelSurprise.H.

Referenced by exportOptions(), and start1().

#### 10.263.4.9 **ModelParam<uint> SingleChannelSurprise::itsSQlen** [protected]

number of surprise maps

Definition at line 75 of file SingleChannelSurprise.H.

Referenced by exportOptions(), and start1().

#### 10.263.4.10 **ModelParam<double> SingleChannelSurprise::itsUpdateFac** [protected]

update factor

Definition at line 76 of file SingleChannelSurprise.H.

Referenced by exportOptions(), getSurpriseMap(), and start1().

The documentation for this class was generated from the following files:

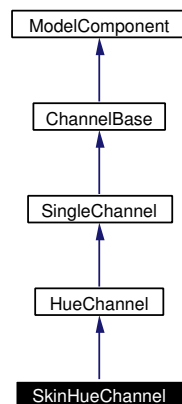
- [SingleChannelSurprise.H](#)
- [SingleChannelSurprise.C](#)

## 10.264 SkinHueChannel Class Reference

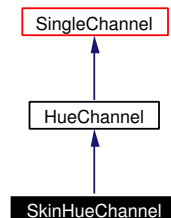
A channel for detecting skin color.

```
#include <Channels.H>
```

Inheritance diagram for SkinHueChannel:



Collaboration diagram for SkinHueChannel:



### Public Member Functions

- [SkinHueChannel](#) ([ModelManager](#) &mgr)

*Constructor.*

- virtual [~SkinHueChannel](#) ()

*virtual destructor, for inheritance*

### 10.264.1 Detailed Description

A channel for detecting skin color.

This implements a [HueChannel](#) with special values for face color derived from a statistical analysis of ~8000 faces in color images.

Definition at line 842 of file Channels.H.

### 10.264.2 Constructor & Destructor Documentation

#### 10.264.2.1 SkinHueChannel::SkinHueChannel ([ModelManager](#) & mgr)

Constructor.

Definition at line 3107 of file Channels.C.

#### 10.264.2.2 SkinHueChannel::~SkinHueChannel () [virtual]

virtual destructor, for inheritance

Definition at line 3119 of file Channels.C.

The documentation for this class was generated from the following files:

- [Channels.H](#)
- [Channels.C](#)

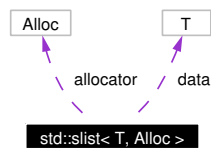


## 10.265 std::slist< T, Alloc > Class Template Reference

The STL slist container class. Use with `#include` .

`#include <dummySTL.H>`

Collaboration diagram for std::slist< T, Alloc >:



### Public Attributes

- T **data**
- Alloc **allocator**

#### 10.265.1 Detailed Description

**template<class T, class Alloc> class std::slist< T, Alloc >**

The STL slist container class. Use with `#include` .

See [SGI's documentation](#) of std::slist.

Definition at line 31 of file dummySTL.H.

The documentation for this class was generated from the following file:

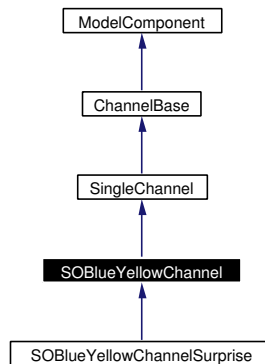
- [dummySTL.H](#)

## 10.266 SOBlueYellowChannel Class Reference

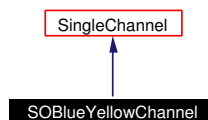
Single Opponent [BlueYellowChannel](#) member definitions:.

```
#include <Channels.H>
```

Inheritance diagram for SOBlueYellowChannel:



Collaboration diagram for SOBlueYellowChannel:



### Public Member Functions

- [SOBlueYellowChannel](#) ([ModelManager](#) &mgr)  
*Construct with standard params.*
- virtual [~SOBlueYellowChannel](#) ()  
*Destructor.*

### 10.266.1 Detailed Description

Single Opponent [BlueYellowChannel](#) member definitions:.

Definition at line 161 of file `Channels.H`.

## 10.266.2 Constructor & Destructor Documentation

### 10.266.2.1 SOBlueYellowChannel::SOBlueYellowChannel ([ModelManager](#) & *mgr*)

Construct with standard params.

Definition at line 204 of file Channels.C.

References [SOBY](#), and [SingleChannel::useSingleOpponence\(\)](#).

### 10.266.2.2 SOBlueYellowChannel::~SOBlueYellowChannel () [virtual]

Destructor.

Definition at line 211 of file Channels.C.

The documentation for this class was generated from the following files:

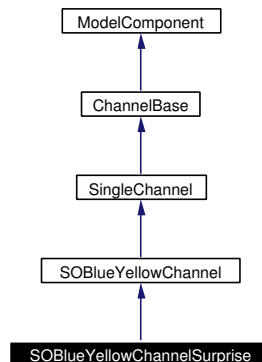
- [Channels.H](#)
- [Channels.C](#)

## 10.267 SOBlueYellowChannelSurprise Class Reference

Single Opponent [BlueYellowChannelSurprise](#) member definitions:.

```
#include <ChannelsSurprise.H>
```

Inheritance diagram for SOBlueYellowChannelSurprise:



Collaboration diagram for SOBlueYellowChannelSurprise:

### Public Member Functions

- [SOBlueYellowChannelSurprise](#) ([ModelManager](#) &mgr)  
*Construct with standard params.*
- virtual [~SOBlueYellowChannelSurprise](#) ()  
*Destructor.*
- virtual [Image](#)< float > [computeSubmap](#) (const [uint](#) index) const  
*[computeSubmap\(\)](#) is special and will return surprise*

### Protected Attributes

- [SharedPtr](#)< [SingleChannelSurprise](#) > itsSCS  
*our surprise computer*

## 10.267.1 Detailed Description

Single Opponent [BlueYellowChannelSurprise](#) member definitions:.

Definition at line 159 of file ChannelsSurprise.H.

## 10.267.2 Constructor & Destructor Documentation

### 10.267.2.1 [SOBlueYellowChannelSurprise::SOBlueYellowChannelSurprise](#) ([ModelManager](#) & *mgr*)

Construct with standard params.

Definition at line 154 of file ChannelsSurprise.C.

References [ModelComponent::addSubComponent\(\)](#).

### 10.267.2.2 [SOBlueYellowChannelSurprise::~~SOBlueYellowChannelSurprise](#) () [virtual]

Destructor.

Definition at line 160 of file ChannelsSurprise.C.

## 10.267.3 Member Function Documentation

### 10.267.3.1 [Image](#)< float > [SOBlueYellowChannelSurprise::computeSubmap](#) (const [uint](#) *index*) const [virtual]

[computeSubmap\(\)](#) is special and will return surprise

Reimplemented from [SingleChannel](#).

Definition at line 163 of file ChannelsSurprise.C.

References [uint](#).

## 10.267.4 Member Data Documentation

### 10.267.4.1 [SharedPtr](#)<[SingleChannelSurprise](#)> [SOBlueYellowChannelSurprise::itsSCS](#) [protected]

our surprise computer

Definition at line 172 of file ChannelsSurprise.H.

The documentation for this class was generated from the following files:

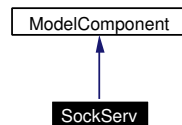
- [ChannelsSurprise.H](#)
- ChannelsSurprise.C

## 10.268 SockServ Class Reference

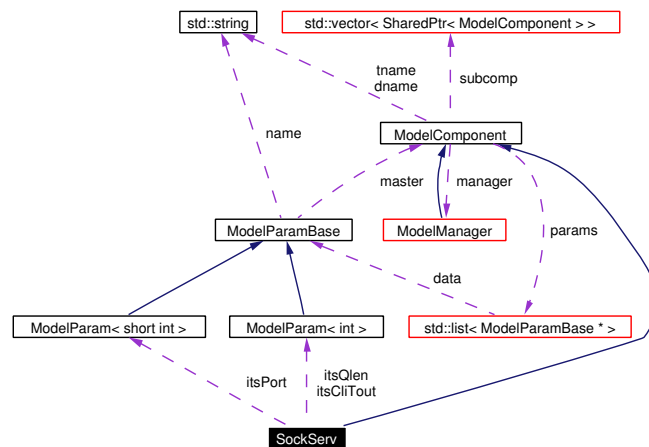
This is a multi-socket connection server.

```
#include <SockServ.H>
```

Inheritance diagram for SockServ:



Collaboration diagram for SockServ:



### Public Member Functions

- **SockServ** (**ModelManager** &mgr, const **std::string** &descrName="Socket Server", const **std::string** &tagName="SockServ")  
*SockServ constructor.*
- **~SockServ** ()  
*SockServ destructor.*
- virtual void **exportOptions** (const int whatToExport, const bool recurse=true)  
*Export some of our **ModelParam** parameters as command-line-options.*

- int **check** (const int stimeout, const int utimeout)  
*check what's going on*
- int **getNewClient** ()  
*zero if no more new clients, else client fd*
- int **getReadClient** ()  
*zero if no more new clients, else client fd*
- int **getWriteClient** ()  
*zero if no more new clients, else client fd*
- int **getErrorClient** ()  
*zero if no more new clients, else client fd*
- in\_addr\_t **getClientIP** (const int clifd) const  
*get IP address of a client*
- short int **getClientPort** (const int clifd) const  
*get port of a client*
- int **getRWClient** ()  
*all wclients first, then all rclients*
- void **disconnect** (const int client)  
*disconnect a client*
- bool **addUserClient** (void \*data)  
*external users can also add their own new clients:*
- void \* **getNewUserClient** ()
- bool **monitorRead** (const int client, const bool startstop=true)  
*what should we monitor for, for a given connection:*
- bool **monitorWrite** (const int client, const bool startstop=true)  
*what should we monitor for, for a given connection:*
- bool **monitorError** (const int client, const bool startstop=true)  
*what should we monitor for, for a given connection:*
- bool **noTimeOut** (const int client)



- void **resetTimeOut** (const int client)
- void **addClient** (const int client)
- void **deleteClient** (const int client)

## Protected Attributes

- [ModelParam](#)< short int > [itsPort](#)  
*My port to listen to.*
- [ModelParam](#)< std::string > [itsServ](#)  
*Name of service in /etc/services.*
- [ModelParam](#)< std::string > [itsServType](#)  
*Type of service in /etc/services.*
- [ModelParam](#)< int > [itsCliTout](#)  
*Timeout on idel clients (in s).*
- [ModelParam](#)< int > [itsQlen](#)  
*Length of listen queue.*

### 10.268.1 Detailed Description

This is a multi-socket connection server.

This is a multi-socket connection server; it listens onto a master socket and accepts incoming connections; it then can monitor all open connections for read, write and/or error, all in a single call to [check\(\)](#), using a non-blocking select().

Classes using SockServ should call [check\(\)](#) to determine which connections require attention. Subsequently, they should take appropriate action on the connections that are active, using [getXXXclient\(\)](#). In addition to the connections that are automatically open by SockServ on [accept\(\)](#) on the master socket, users may tell SockServ to monitor additional (already open) connections, by adding them to SockServ's monitor list using [addUserClient\(\)](#). By default, connections time out after CLITIMEOUT time of inactivity. This can be disabled using [noTimeOut\(\)](#).

Definition at line 71 of file SockServ.H.

## 10.268.2 Constructor & Destructor Documentation

### 10.268.2.1 SockServ::SockServ ([ModelManager](#) & *mgr*, const [std::string](#) & *descrName* = "Socket Server", const [std::string](#) & *tagName* = "SockServ")

SockServ constructor.

Definition at line 67 of file SockServ.C.

References SOCKSERV\_UNCONFIGURED.

### 10.268.2.2 SockServ::~~SockServ ()

SockServ destructor.

Definition at line 78 of file SockServ.C.

## 10.268.3 Member Function Documentation

### 10.268.3.1 bool SockServ::addUserClient (void \* *data*)

external users can also add their own new clients:

Definition at line 298 of file SockServ.C.

References MAXNBNUCLI.

### 10.268.3.2 int SockServ::check (const int *stimeout*, const int *utimeout*)

check what's going on

Definition at line 157 of file SockServ.C.

References CLITIMEOUT, disconnect(), and SOCKSERV\_IDLE.

### 10.268.3.3 void SockServ::disconnect (const int *client*)

disconnect a client

Definition at line 283 of file SockServ.C.

Referenced by check().

**10.268.3.4 void SockServ::exportOptions (const int *whatToExport*, const bool *recurse* = true) [virtual]**

Export some of our [ModelParam](#) parameters as command-line-options.

See the base function in [ModelComponent.H](#) for info.

Reimplemented from [ModelComponent](#).

Definition at line 82 of file SockServ.C.

References [ModelComponent::exportOptions\(\)](#), [OPTEXP\\_CORE](#), and [ModelManager::requestOption\(\)](#).

**10.268.3.5 in\_addr\_t SockServ::getClientIP (const int *clifd*) const**

get IP address of a client

Definition at line 257 of file SockServ.C.

**10.268.3.6 short int SockServ::getClientPort (const int *clifd*) const**

get port of a client

Definition at line 266 of file SockServ.C.

**10.268.3.7 int SockServ::getErrorClient ()**

zero if no more new clients, else client fd

Definition at line 253 of file SockServ.C.

**10.268.3.8 int SockServ::getNewClient ()**

zero if no more new clients, else client fd

Definition at line 241 of file SockServ.C.

**10.268.3.9 int SockServ::getReadClient ()**

zero if no more new clients, else client fd

Definition at line 245 of file SockServ.C.

**10.268.3.10 int SockServ::getRWClient ()**

all wclients first, then all rclients

Definition at line 275 of file SockServ.C.

**10.268.3.11 int SockServ::getWriteClient ()**

zero if no more new clients, else client fd

Definition at line 249 of file SockServ.C.

**10.268.3.12 bool SockServ::monitorError (const int *client*, const bool *startstop* = true)**

what should we monitor for, for a given connection:

Definition at line 355 of file SockServ.C.

**10.268.3.13 bool SockServ::monitorRead (const int *client*, const bool *startstop* = true)**

what should we monitor for, for a given connection:

Definition at line 335 of file SockServ.C.

**10.268.3.14 bool SockServ::monitorWrite (const int *client*, const bool *startstop* = true)**

what should we monitor for, for a given connection:

Definition at line 345 of file SockServ.C.

**10.268.4 Member Data Documentation****10.268.4.1 [ModelParam<int>](#) [SockServ::itsCliTout](#) [protected]**

Timeout on idel clients (in s).

Definition at line 137 of file SockServ.H.

**10.268.4.2 [ModelParam<short int>](#) [SockServ::itsPort](#) [protected]**

My port to listen to.

Definition at line 134 of file SockServ.H.

#### 10.268.4.3 [ModelParam<int>](#) [SockServ::itsQlen](#) [protected]

Length of listen queue.

Definition at line 138 of file SockServ.H.

#### 10.268.4.4 [ModelParam<std::string>](#) [SockServ::itsServ](#) [protected]

Name of service in /etc/services.

Definition at line 135 of file SockServ.H.

#### 10.268.4.5 [ModelParam<std::string>](#) [SockServ::itsServType](#) [protected]

Type of service in /etc/services.

Definition at line 136 of file SockServ.H.

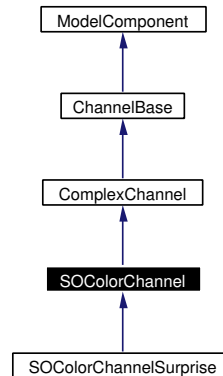
The documentation for this class was generated from the following files:

- [SockServ.H](#)
- [SockServ.C](#)

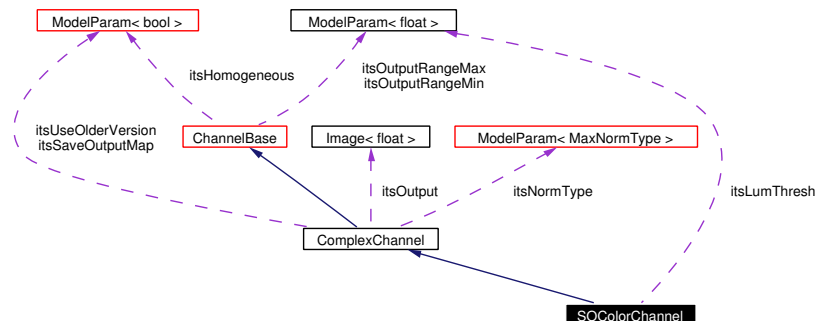
## 10.269 SOColorChannel Class Reference

```
#include <Channels.H>
```

Inheritance diagram for SOColorChannel:



Collaboration diagram for SOColorChannel:



### Public Member Functions

- `SOColorChannel (ModelManager &mgr)`  
*Construct with standard params.*
- `virtual ~SOColorChannel ()`  
*Destructor.*
- `virtual SORedGreenChannel & rg () const`

Get access the single opponent [RedGreenChannel](#) subchannel.

- virtual [SOGreenRedChannel](#) & [gr](#) () const  
Get access the single opponent [GreenRedChannel](#) subchannel.
- virtual [SOBlueYellowChannel](#) & [by](#) () const  
Get access to the single opponent [BlueYellowChannel](#) subchannel.
- virtual [SOYellowBlueChannel](#) & [yb](#) () const  
Get access to the single opponent [YellowBlueChannel](#) subchannel.
- virtual [RedChannel](#) & [r](#) () const  
Get access the [RedChannel](#) subchannel.
- virtual [GreenChannel](#) & [g](#) () const  
Get access the [GreenChannel](#) subchannel.
- virtual [BlueChannel](#) & [b](#) () const  
Get access to the [BlueChannel](#) subchannel.
- virtual [YellowChannel](#) & [y](#) () const  
Get access to the [YellowChannel](#) subchannel.

## Protected Member Functions

- virtual void [doInput](#) (const [Image](#)< [PixRGB](#)< float > > \*colimg, const [Image](#)< float > \*, const double t, const [Image](#)< [byte](#) > &clipMask)  
*SOColorChannel requires only color input.*

## Protected Attributes

- [ModelParam](#)< float > [itsLumThresh](#)  
Luminance threshold (see [getRGBY\(\)](#)).
- [uint](#) [itsRGIdx](#)  
subComponent indices for rg, gr, by, yb
- [uint](#) [itsGRIdx](#)  
subComponent indices for rg, gr, by, yb

- [uint itsBYidx](#)  
*subComponent indices for rg, gr, by, yb*
- [uint itsYBidx](#)  
*subComponent indices for rg, gr, by, yb*
- [uint itsRidx](#)  
*subComponent indices for r, g, b, y*
- [uint itsGidx](#)  
*subComponent indices for r, g, b, y*
- [uint itsBidx](#)  
*subComponent indices for r, g, b, y*
- [uint itsYidx](#)  
*subComponent indices for r, g, b, y*

### 10.269.1 Detailed Description

This is an example of an heterogeneous complex channel where the various subchannels have distinct visual features. Thus, the [JetSpec](#) representation for such complex channel will be a series of independent single-channel representations, one for each subchannel.

Definition at line 973 of file Channels.H.

### 10.269.2 Constructor & Destructor Documentation

#### 10.269.2.1 SOColorChannel::SOColorChannel ([ModelManager](#) & mgr)

Construct with standard params.

Definition at line 895 of file Channels.C.

References [ComplexChannel::addSubChan\(\)](#), [b\(\)](#), [g\(\)](#), [itsBidx](#), [itsGidx](#), [itsGRidx](#), [itsRidx](#), [itsYBidx](#), [itsYidx](#), [makeShared\(\)](#), [r\(\)](#), [ChannelBase::setTotalWeight\(\)](#), [SO-COLOR](#), and [y\(\)](#).

#### 10.269.2.2 SOColorChannel::~~SOColorChannel () [virtual]

Destructor.

Definition at line 981 of file Channels.C.



## 10.269.3 Member Function Documentation

### 10.269.3.1 [BlueChannel](#) & SOColorChannel::b () const [virtual]

Get access to the [BlueChannel](#) subchannel.

Reimplemented in [SOColorChannelSurprise](#).

Definition at line 966 of file Channels.C.

References itsBidx, and ComplexChannel::subChan().

Referenced by doInput(), and SOColorChannel().

### 10.269.3.2 [SOBlueYellowChannel](#) & SOColorChannel::by () const [virtual]

Get access to the single opponent [BlueYellowChannel](#) subchannel.

Reimplemented in [SOColorChannelSurprise](#).

Definition at line 935 of file Channels.C.

References ComplexChannel::subChan().

Referenced by doInput().

### 10.269.3.3 void SOColorChannel::doInput (const [Image](#)< [PixRGB](#)< float > > \* *colimg*, const [Image](#)< float > \*, const double *t*, const [Image](#)< byte > & *clipMask*) [protected, virtual]

SOColorChannel requires only color input.

Implements [ChannelBase](#).

Definition at line 985 of file Channels.C.

References ASSERT, b(), by(), g(), getRGBY(), ModelParam< float >::getVal(), gr(), ChannelBase::input(), itsBidx, itsGidx, itsRidx, itsYidx, r(), rg(), SingleChannel::singleOpponentInput(), ComplexChannel::subChan(), y(), and yb().

### 10.269.3.4 [GreenChannel](#) & SOColorChannel::g () const [virtual]

Get access the [GreenChannel](#) subchannel.

Reimplemented in [SOColorChannelSurprise](#).

Definition at line 958 of file Channels.C.

References itsGidx, and ComplexChannel::subChan().

Referenced by `doInput()`, and `SOColorChannel()`.

#### **10.269.3.5** [SOGreenRedChannel](#) & `SOColorChannel::gr () const` [virtual]

Get access the single opponent `GreenRedChannel` subchannel.

Reimplemented in [SOColorChannelSurprise](#).

Definition at line 927 of file `Channels.C`.

References `itsGRidx`, and `ComplexChannel::subChan()`.

Referenced by `doInput()`.

#### **10.269.3.6** [RedChannel](#) & `SOColorChannel::r () const` [virtual]

Get access the [RedChannel](#) subchannel.

Reimplemented in [SOColorChannelSurprise](#).

Definition at line 950 of file `Channels.C`.

References `itsRidx`, and `ComplexChannel::subChan()`.

Referenced by `doInput()`, and `SOColorChannel()`.

#### **10.269.3.7** [SORedGreenChannel](#) & `SOColorChannel::rg () const` [virtual]

Get access the single opponent [RedGreenChannel](#) subchannel.

Reimplemented in [SOColorChannelSurprise](#).

Definition at line 919 of file `Channels.C`.

References `ComplexChannel::subChan()`.

Referenced by `doInput()`.

#### **10.269.3.8** [YellowChannel](#) & `SOColorChannel::y () const` [virtual]

Get access to the [YellowChannel](#) subchannel.

Reimplemented in [SOColorChannelSurprise](#).

Definition at line 974 of file `Channels.C`.

References `itsYidx`, and `ComplexChannel::subChan()`.

Referenced by `doInput()`, and `SOColorChannel()`.

### 10.269.3.9 [SOYellowBlueChannel](#) & [SOColorChannel::yb \(\) const](#) [virtual]

Get access to the single opponent YellowBlueChannel subchannel.

Reimplemented in [SOColorChannelSurprise](#).

Definition at line 943 of file Channels.C.

References [itsYBidx](#), and [ComplexChannel::subChan\(\)](#).

Referenced by [doInput\(\)](#).

## 10.269.4 Member Data Documentation

### 10.269.4.1 [uint SOColorChannel::itsBidx](#) [protected]

subComponent indices for r, g, b, y

Definition at line 1018 of file Channels.H.

Referenced by [b\(\)](#), [doInput\(\)](#), and [SOColorChannel\(\)](#).

### 10.269.4.2 [uint SOColorChannel::itsBYidx](#) [protected]

subComponent indices for rg, gr, by, yb

Definition at line 1015 of file Channels.H.

### 10.269.4.3 [uint SOColorChannel::itsGidx](#) [protected]

subComponent indices for r, g, b, y

Definition at line 1018 of file Channels.H.

Referenced by [doInput\(\)](#), [g\(\)](#), and [SOColorChannel\(\)](#).

### 10.269.4.4 [uint SOColorChannel::itsGRidx](#) [protected]

subComponent indices for rg, gr, by, yb

Definition at line 1015 of file Channels.H.

Referenced by [gr\(\)](#), and [SOColorChannel\(\)](#).

**10.269.4.5** [ModelParam<float> SOColorChannel::itsLumThresh](#)  
[protected]

Luminance threshold (see [getRGBY\(\)](#)).

Definition at line 1007 of file Channels.H.

**10.269.4.6** [uint SOColorChannel::itsRGIdx](#) [protected]

subComponent indices for rg, gr, by, yb

Definition at line 1015 of file Channels.H.

**10.269.4.7** [uint SOColorChannel::itsRidx](#) [protected]

subComponent indices for r, g, b, y

Definition at line 1018 of file Channels.H.

Referenced by [doInput\(\)](#), [r\(\)](#), and [SOColorChannel\(\)](#).

**10.269.4.8** [uint SOColorChannel::itsYBidx](#) [protected]

subComponent indices for rg, gr, by, yb

Definition at line 1015 of file Channels.H.

Referenced by [SOColorChannel\(\)](#), and [yb\(\)](#).

**10.269.4.9** [uint SOColorChannel::itsYidx](#) [protected]

subComponent indices for r, g, b, y

Definition at line 1018 of file Channels.H.

Referenced by [doInput\(\)](#), [SOColorChannel\(\)](#), and [y\(\)](#).

The documentation for this class was generated from the following files:

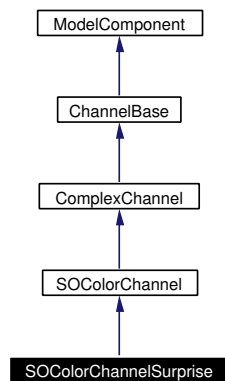
- [Channels.H](#)
- [Channels.C](#)

## 10.270 SOColorChannelSurprise Class Reference

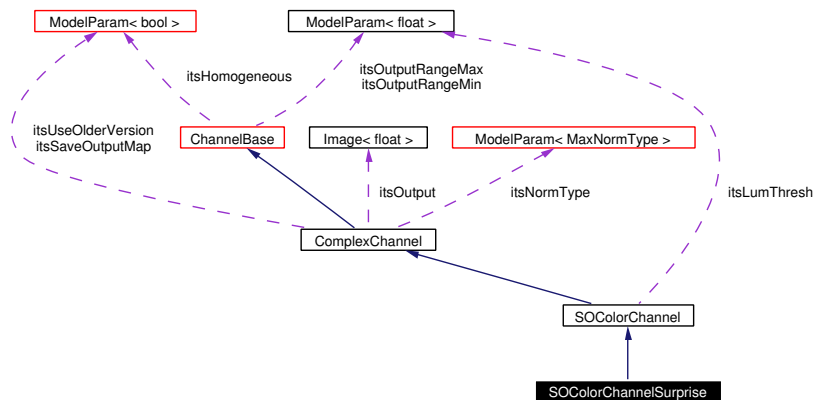
A single opponent color channel class that combines r/g, g/r, b/y, etc.

```
#include <ChannelsSurprise.H>
```

Inheritance diagram for SOColorChannelSurprise:



Collaboration diagram for SOColorChannelSurprise:



### Public Member Functions

- **SOColorChannelSurprise** (**ModelManager** &mgr)

*Construct with standard params.*

- virtual [~SOColorChannelSurprise \(\)](#)  
*Destructor.*
- virtual [SOREdGreenChannelSurprise & rg \(\)](#) const  
*Get access the single opponent [RedGreenChannelSurprise](#) subchannel.*
- virtual [SOGreenRedChannelSurprise & gr \(\)](#) const  
*Get access the single opponent [GreenRedChannelSurprise](#) subchannel.*
- virtual [SOBlueYellowChannelSurprise & by \(\)](#) const  
*Get access to the single opponent [BlueYellowChannelSurprise](#) subchannel.*
- virtual [SOYellowBlueChannelSurprise & yb \(\)](#) const  
*Get access to the single opponent [YellowBlueChannelSurprise](#) subchannel.*
- virtual [RedChannelSurprise & r \(\)](#) const  
*Get access the [RedChannelSurprise](#) subchannel.*
- virtual [GreenChannelSurprise & g \(\)](#) const  
*Get access the [GreenChannelSurprise](#) subchannel.*
- virtual [BlueChannelSurprise & b \(\)](#) const  
*Get access to the [BlueChannelSurprise](#) subchannel.*
- virtual [YellowChannelSurprise & y \(\)](#) const  
*Get access to the [YellowChannelSurprise](#) subchannel.*

### 10.270.1 Detailed Description

A single opponent color channel class that combines r/g, g/r, b/y, etc.

Definition at line 470 of file ChannelsSurprise.H.

### 10.270.2 Constructor & Destructor Documentation

#### 10.270.2.1 [SOColorChannelSurprise::SOColorChannelSurprise \(ModelManager & mgr\)](#)

Construct with standard params.

Definition at line 473 of file ChannelsSurprise.C.

References `ComplexChannel::addSubChan()`, `b()`, `g()`, `makeShared()`, `r()`, `ModelComponent::removeAllSubComponents()`, `ChannelBase::setHomogeneous()`, `ChannelBase::setTotalWeight()`, and `y()`.

#### 10.270.2.2 `SOColorChannelSurprise::~~SOColorChannelSurprise ()` [virtual]

Destructor.

Definition at line 501 of file `ChannelsSurprise.C`.

### 10.270.3 Member Function Documentation

#### 10.270.3.1 `BlueChannelSurprise & SOColorChannelSurprise::b () const` [virtual]

Get access to the `BlueChannelSurprise` subchannel.

Reimplemented from `SOColorChannel`.

Definition at line 529 of file `ChannelsSurprise.C`.

References `ComplexChannel::subChan()`.

Referenced by `SOColorChannelSurprise()`.

#### 10.270.3.2 `SOBlueYellowChannelSurprise & SOColorChannelSurprise::by () const` [virtual]

Get access to the single opponent `BlueYellowChannelSurprise` subchannel.

Reimplemented from `SOColorChannel`.

Definition at line 513 of file `ChannelsSurprise.C`.

References `ComplexChannel::subChan()`.

#### 10.270.3.3 `GreenChannelSurprise & SOColorChannelSurprise::g () const` [virtual]

Get access the `GreenChannelSurprise` subchannel.

Reimplemented from `SOColorChannel`.

Definition at line 525 of file `ChannelsSurprise.C`.

References `ComplexChannel::subChan()`.

Referenced by `SOColorChannelSurprise()`.

#### 10.270.3.4 [SOGreenRedChannelSurprise](#) & [SOColorChannelSurprise::gr \(\)](#) `const [virtual]`

Get access the single opponent GreenRedChannelSurprise subchannel.

Reimplemented from [SOColorChannel](#).

Definition at line 509 of file ChannelsSurprise.C.

References [ComplexChannel::subChan\(\)](#).

#### 10.270.3.5 [RedChannelSurprise](#) & [SOColorChannelSurprise::r \(\)](#) `const [virtual]`

Get access the [RedChannelSurprise](#) subchannel.

Reimplemented from [SOColorChannel](#).

Definition at line 521 of file ChannelsSurprise.C.

References [ComplexChannel::subChan\(\)](#).

Referenced by [SOColorChannelSurprise\(\)](#).

#### 10.270.3.6 [SORedGreenChannelSurprise](#) & [SOColorChannelSurprise::rg \(\)](#) `const [virtual]`

Get access the single opponent [RedGreenChannelSurprise](#) subchannel.

Reimplemented from [SOColorChannel](#).

Definition at line 505 of file ChannelsSurprise.C.

References [ComplexChannel::subChan\(\)](#).

#### 10.270.3.7 [YellowChannelSurprise](#) & [SOColorChannelSurprise::y \(\)](#) `const [virtual]`

Get access to the [YellowChannelSurprise](#) subchannel.

Reimplemented from [SOColorChannel](#).

Definition at line 533 of file ChannelsSurprise.C.

References [ComplexChannel::subChan\(\)](#).

Referenced by [SOColorChannelSurprise\(\)](#).



### 10.270.3.8 [SOYellowBlueChannelSurprise](#) & SOColorChannelSurprise::yb () const [virtual]

Get access to the single opponent YellowBlueChannelSurprise subchannel.

Reimplemented from [SOColorChannel](#).

Definition at line 517 of file ChannelsSurprise.C.

References [ComplexChannel::subChan\(\)](#).

The documentation for this class was generated from the following files:

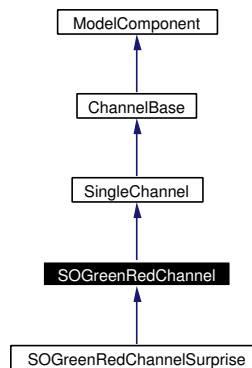
- [ChannelsSurprise.H](#)
- ChannelsSurprise.C

## 10.271 SOGreenRedChannel Class Reference

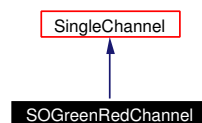
Single Opponent GreenRedChannel member definitions:.

```
#include <Channels.H>
```

Inheritance diagram for SOGreenRedChannel:



Collaboration diagram for SOGreenRedChannel:



### Public Member Functions

- [SOGreenRedChannel](#) ([ModelManager](#) &mgr)  
*Construct with standard params.*
- virtual [~SOGreenRedChannel](#) ()  
*Destructor.*

#### 10.271.1 Detailed Description

Single Opponent GreenRedChannel member definitions:.

Definition at line 149 of file `Channels.H`.

## 10.271.2 Constructor & Destructor Documentation

### 10.271.2.1 SOGreenRedChannel::SOGreenRedChannel ([ModelManager](#) & *mgr*)

Construct with standard params.

Definition at line 191 of file Channels.C.

References [SOGR](#), and [SingleChannel::useSingleOpponence\(\)](#).

### 10.271.2.2 SOGreenRedChannel::~~[SOGreenRedChannel](#) () [virtual]

Destructor.

Definition at line 198 of file Channels.C.

The documentation for this class was generated from the following files:

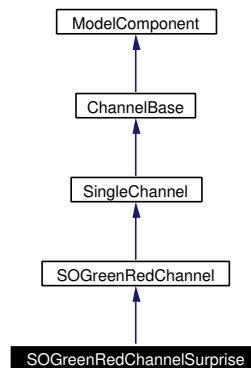
- [Channels.H](#)
- [Channels.C](#)

## 10.272 SOGreenRedChannelSurprise Class Reference

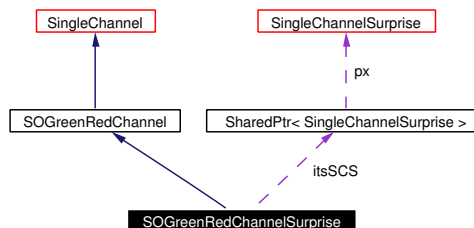
Single Opponent GreenRedChannelSurprise member definitions:.

```
#include <ChannelsSurprise.H>
```

Inheritance diagram for SOGreenRedChannelSurprise:



Collaboration diagram for SOGreenRedChannelSurprise:



### Public Member Functions

- **SOGreenRedChannelSurprise** (**ModelManager** &mgr)  
*Construct with standard params.*
- virtual **~SOGreenRedChannelSurprise** ()  
*Destructor.*
- virtual **Image**< float > **computeSubmap** (const **uint** index) const  
*computeSubmap() is special and will return surprise*

## Protected Attributes

- [SharedPtr](#)< [SingleChannelSurprise](#) > [itsSCS](#)  
*our surprise computer*

### 10.272.1 Detailed Description

Single Opponent GreenRedChannelSurprise member definitions:.  
Definition at line 141 of file ChannelsSurprise.H.

### 10.272.2 Constructor & Destructor Documentation

#### 10.272.2.1 SOGreenRedChannelSurprise::SOGreenRedChannelSurprise ([ModelManager](#) & *mgr*)

Construct with standard params.  
Definition at line 139 of file ChannelsSurprise.C.  
References [ModelComponent::addSubComponent\(\)](#).

#### 10.272.2.2 SOGreenRedChannelSurprise::~~SOGreenRedChannelSurprise () [virtual]

Destructor.  
Definition at line 145 of file ChannelsSurprise.C.

### 10.272.3 Member Function Documentation

#### 10.272.3.1 [Image](#)< float > SOGreenRedChannelSurprise::computeSubmap (const [uint](#) *index*) const [virtual]

[computeSubmap\(\)](#) is special and will return surprise  
Reimplemented from [SingleChannel](#).  
Definition at line 148 of file ChannelsSurprise.C.  
References [uint](#).

## 10.272.4 Member Data Documentation

### 10.272.4.1 [SharedPtr<SingleChannelSurprise>](#) [SOGreenRedChannelSurprise::itsSCS](#) [protected]

our surprise computer

Definition at line 154 of file ChannelsSurprise.H.

The documentation for this class was generated from the following files:

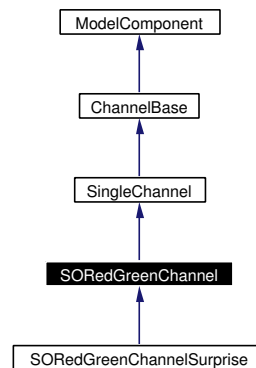
- [ChannelsSurprise.H](#)
- ChannelsSurprise.C

## 10.273 SORedGreenChannel Class Reference

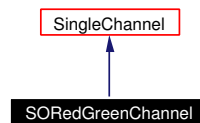
Single Opponent [RedGreenChannel](#) member definitions:.

```
#include <Channels.H>
```

Inheritance diagram for SORedGreenChannel:



Collaboration diagram for SORedGreenChannel:



### Public Member Functions

- [SORedGreenChannel](#) ([ModelManager](#) &mgr)  
*Construct with standard params.*
- virtual [~SORedGreenChannel](#) ()  
*Destructor.*

### 10.273.1 Detailed Description

Single Opponent [RedGreenChannel](#) member definitions:.

Definition at line 137 of file `Channels.H`.

## 10.273.2 Constructor & Destructor Documentation

### 10.273.2.1 `SORedGreenChannel::SORedGreenChannel` ([ModelManager](#) & *mgr*)

Construct with standard params.

Definition at line 178 of file `Channels.C`.

References `SORG`, and `SingleChannel::useSingleOpponence()`.

### 10.273.2.2 `SORedGreenChannel::~~SORedGreenChannel` () [virtual]

Destructor.

Definition at line 185 of file `Channels.C`.

The documentation for this class was generated from the following files:

- [Channels.H](#)
- [Channels.C](#)

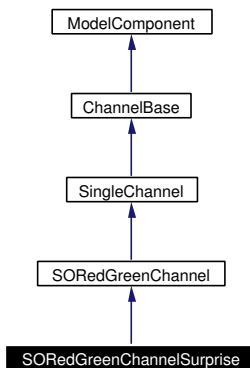


## 10.274 SORedGreenChannelSurprise Class Reference

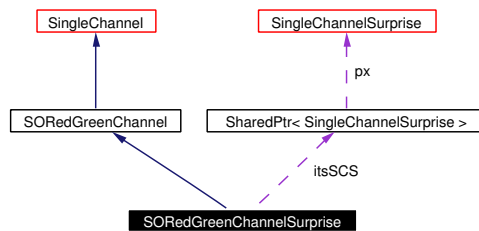
Single Opponent [RedGreenChannelSurprise](#) member definitions:.

```
#include <ChannelsSurprise.H>
```

Inheritance diagram for SORedGreenChannelSurprise:



Collaboration diagram for SORedGreenChannelSurprise:



### Public Member Functions

- [SORedGreenChannelSurprise](#) ([ModelManager](#) &mgr)  
*Construct with standard params.*
- virtual [~SORedGreenChannelSurprise](#) ()  
*Destructor.*
- virtual [Image< float >](#) [computeSubmap](#) (const [uint](#) index) const  
*[computeSubmap\(\)](#) is special and will return surprise*

## Protected Attributes

- [SharedPtr](#)< [SingleChannelSurprise](#) > [itsSCS](#)  
*our surprise computer*

### 10.274.1 Detailed Description

Single Opponent [RedGreenChannelSurprise](#) member definitions:.  
Definition at line 123 of file ChannelsSurprise.H.

### 10.274.2 Constructor & Destructor Documentation

#### 10.274.2.1 [SRedGreenChannelSurprise::SRedGreenChannelSurprise](#) ([ModelManager](#) & *mgr*)

Construct with standard params.  
Definition at line 124 of file ChannelsSurprise.C.  
References [ModelComponent::addSubComponent\(\)](#).

#### 10.274.2.2 [SRedGreenChannelSurprise::~~SRedGreenChannelSurprise](#) () [virtual]

Destructor.  
Definition at line 130 of file ChannelsSurprise.C.

### 10.274.3 Member Function Documentation

#### 10.274.3.1 [Image](#)< float > [SRedGreenChannelSurprise::computeSubmap](#) (const [uint](#) *index*) const [virtual]

[computeSubmap\(\)](#) is special and will return surprise  
Reimplemented from [SingleChannel](#).  
Definition at line 133 of file ChannelsSurprise.C.  
References [uint](#).

## 10.274.4 Member Data Documentation

### 10.274.4.1 [SharedPtr<SingleChannelSurprise> SORedGreenChannelSurprise::itsSCS](#) [protected]

our surprise computer

Definition at line 136 of file ChannelsSurprise.H.

The documentation for this class was generated from the following files:

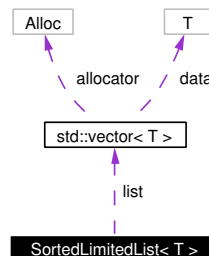
- [ChannelsSurprise.H](#)
- ChannelsSurprise.C

## 10.275 SortedLimitedList< T > Class Template Reference

Maintains a sorted list with a limited number of elements.

```
#include <KDTree.H>
```

Collaboration diagram for SortedLimitedList< T >:



### Public Member Functions

- [SortedLimitedList](#) ()  
*Constructor.*
- [SortedLimitedList](#) (int maxElements)  
*Constructor.*
- int [add](#) (const T &entry, int order=0)  
*Add an entry to the list.*
- void [addUnsorted](#) (const T &entry)  
*Add an entry to the list without yet sorting.*
- const T & [getValue](#) (int index) const  
*Get an entry from the list.*
- int [getSize](#) () const  
*Get the size of the list.*
- int [indexOf](#) (const T &entry) const  
*Get the index of the entry.*
- void [sort](#) (int n, int start=0, const int order=0)

*Sort the list 0=ascending order, 1=descending order.*

- void [remove](#) (int index)  
*Remove the object at the specified index from the list.*
- void [setMaxElements](#) (int value)  
*Change the max number of elements.*
- const [std::vector](#)< T > & [toVector](#) () const  
*Convert to vector.*

### 10.275.1 Detailed Description

**template<class T> class SortedLimitedList< T >**

Maintains a sorted list with a limited number of elements.

Definition at line 87 of file KDTree.H.

### 10.275.2 Constructor & Destructor Documentation

**10.275.2.1** **template<class T> [SortedLimitedList](#)< T >::[SortedLimitedList](#) ()**  
[inline]

Constructor.

Definition at line 137 of file KDTree.H.

**10.275.2.2** **template<class T> [SortedLimitedList](#)< T >::[SortedLimitedList](#)**  
**(int *maxElements*)** [inline]

Constructor.

Definition at line 144 of file KDTree.H.

### 10.275.3 Member Function Documentation

**10.275.3.1** **template<class T> int [SortedLimitedList](#)< T >::[add](#) (const T &**  
***entry*, int *order* = 0)** [inline]

Add an entry to the list.

Definition at line 151 of file KDTree.H.

References SortedLimitedList< T >::indexOf(), SortedLimitedList< T >::sort(), and uint.

**10.275.3.2** `template<class T> void SortedLimitedList< T >::addUnsorted  
(const T & entry) [inline]`

Add an entry to the list without yet sorting.

Definition at line 162 of file KDTree.H.

**10.275.3.3** `template<class T> int SortedLimitedList< T >::getSize () const  
[inline]`

Get the size of the list.

Definition at line 189 of file KDTree.H.

**10.275.3.4** `template<class T> const T & SortedLimitedList< T >::getValue  
(int index) const [inline]`

Get an entry from the list.

Definition at line 182 of file KDTree.H.

**10.275.3.5** `template<class T> int SortedLimitedList< T >::indexOf (const T  
& entry) const [inline]`

Get the index of the entry.

Definition at line 169 of file KDTree.H.

References uint.

Referenced by SortedLimitedList< T >::add().

**10.275.3.6** `template<class T> void SortedLimitedList< T >::remove (int  
index) [inline]`

Remove the object at the specified index from the list.

Definition at line 254 of file KDTree.H.

**10.275.3.7** `template<class T> void SortedLimitedList< T >::setMaxElements  
(int value) [inline]`

Change the max number of elements.

Definition at line 196 of file KDTree.H.

References uint.

**10.275.3.8** `template<class T> void SortedLimitedList< T >::sort (int n, int  
start = 0, const int order = 0) [inline]`

Sort the list 0=ascending order, 1=descending order.

Definition at line 205 of file KDTree.H.

Referenced by SortedLimitedList< T >::add().

**10.275.3.9** `template<class T> const std::vector< T > & SortedLimitedList< T  
>::toVector () const [inline]`

Convert to vector.

Definition at line 261 of file KDTree.H.

The documentation for this class was generated from the following file:

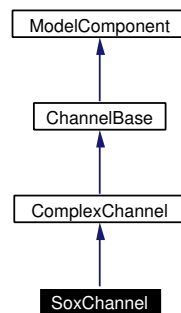
- [KDTree.H](#)

## 10.276 SoxChannel Class Reference

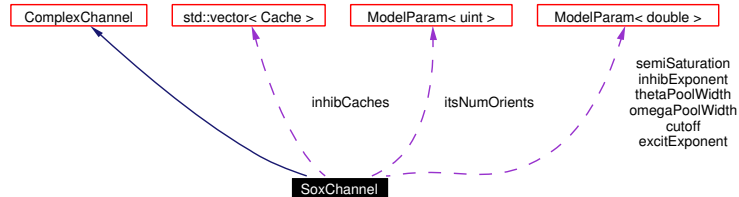
A short-range orientation interactions channel.

```
#include <SoxChannel.h>
```

Inheritance diagram for SoxChannel:



Collaboration diagram for SoxChannel:



### Public Member Functions

- [SoxChannel](#) ([ModelManager](#) &mgr)  
*Construct.*
- virtual [~SoxChannel](#) ()  
*Virtual destructor.*
- virtual void [exportOptions](#) (const int whatToExport, const bool recurse=true)  
*Export some of our [ModelParam](#) parameters as command-line-options.*
- [uint numScales](#) () const  
*Get the number of scales used.*



- `Image< float > getLinearResponse (int ori, int scl)`  
*Get the linear response of the units for a given orientation+scale.*
- `Image< float > getNonlinearResponse (int ori, int scl)`  
*Get the non-linear response of the units for a given orientation+scale.*
- `GaborChannel & subChan (const uint idx) const`  
*Wraps ComplexChannel's `subChan()` and casts the result to `GaborChannel`.*

## Protected Member Functions

- `void buildSubChans ()`  
*(re-)build our subchannels*
- `virtual void doInput (const Image< PixRGB< float > > *, const Image< float > *bwimg, const double t, const Image< byte > &clipMask)`  
*SoxChannel requires only luminance input.*
- `virtual Image< float > combineOutputs ()`  
*Combine all of the nonlinear responses.*
- `virtual void killCaches ()`  
*Clear our cached inhibitory pool maps.*
- `bool setModelParamString (const std::string &name, const std::string &value, const bool recurse=false, const bool warn=true)`  
*Overload so that we can reconfigure when our params get changed.*

## Protected Attributes

- `ModelParam< uint > itsNumOrients`  
*number of Gabor subchannels*
- `ModelParam< double > thetaPoolWidth`
- `ModelParam< double > omegaPoolWidth`
- `ModelParam< double > inhibExponent`
- `ModelParam< double > excitExponent`
- `ModelParam< double > semiSaturation`
- `ModelParam< double > cutoff`

### 10.276.1 Detailed Description

A short-range orientation interactions channel.

This class implements a pool of orientation-tuned units whose responses are shaped by divisive inhibition in proportion to population activity.

Definition at line 46 of file SoxChannel.H.

### 10.276.2 Constructor & Destructor Documentation

#### 10.276.2.1 SoxChannel::SoxChannel ([ModelManager](#) & mgr)

Construct.

Definition at line 51 of file SoxChannel.C.

References [buildSubChans\(\)](#), and [ORI](#).

#### 10.276.2.2 SoxChannel::~~SoxChannel () [virtual]

Virtual destructor.

Definition at line 127 of file SoxChannel.C.

### 10.276.3 Member Function Documentation

#### 10.276.3.1 void SoxChannel::buildSubChans () [protected]

(re-)build our subchannels

Definition at line 67 of file SoxChannel.C.

References [ComplexChannel::addSubChan\(\)](#), [ModelComponent::exportOptions\(\)](#), [ModelParam< uint >::getVal\(\)](#), [makeShared\(\)](#), [OPTEXP\\_NONE](#), [ModelComponent::removeAllSubComponents\(\)](#), [ChannelBase::setHomogeneous\(\)](#), and [uint](#).

Referenced by [setModelParamString\(\)](#), and [SoxChannel\(\)](#).

#### 10.276.3.2 [Image< float >](#) SoxChannel::combineOutputs () [protected, virtual]

Combine all of the nonlinear responses.

Reimplemented from [ComplexChannel](#).

Definition at line 235 of file SoxChannel.C.

References [ComplexChannel::getMapDims\(\)](#), [getNonlinearResponse\(\)](#), [ModelParam< MaxNormType >::getVal\(\)](#), [maxNormalize\(\)](#), [MAXNORMMAX](#), [MAXNORMMIN](#), [ComplexChannel::numChans\(\)](#), [numScales\(\)](#), [rescale\(\)](#), [uint](#), [VCXNORM\\_MAXNORM](#), and [ZEROS](#).

**10.276.3.3** `void SoxChannel::doInput (const Image< PixRGB< float > > \*,  
const Image< float > \* bwing, const double t, const Image< byte > \*  
& clipMask)` [`protected`, `virtual`]

SoxChannel requires only luminance input.

Implements [ChannelBase](#).

Definition at line 135 of file SoxChannel.C.

References [ASSERT](#), [ChannelBase::input\(\)](#), [ComplexChannel::numChans\(\)](#), [subChan\(\)](#), and [uint](#).

**10.276.3.4** `void SoxChannel::exportOptions (const int whatToExport, const bool  
recurse = true)` [`virtual`]

Export some of our [ModelParam](#) parameters as command-line-options.

See the base function in [ModelComponent.H](#) for info.

Reimplemented from [ComplexChannel](#).

Definition at line 109 of file SoxChannel.C.

References [ComplexChannel::exportOptions\(\)](#), [OPTEXP\\_CORE](#), and [ModelManager::requestOption\(\)](#).

**10.276.3.5** `Image< float > SoxChannel::getLinearResponse \(int ori, int scl\)`

Get the linear response of the units for a given orientation+scale.

Definition at line 150 of file SoxChannel.C.

References [SingleChannel::getImage\(\)](#), and [subChan\(\)](#).

Referenced by [getNonlinearResponse\(\)](#).

**10.276.3.6** `Image< float > SoxChannel::getNonlinearResponse \(int ori, int scl\)`

Get the non-linear response of the units for a given orientation+scale.

This is (roughly speaking) given by a power of the linear response, divided by an inhibitory term that pools over all orientations and scales.

Definition at line 156 of file SoxChannel.C.

References `Image< T >::addWeighted()`, `GaborChannel::angle()`, `Image< T >::clear()`, `Image< T >::getDims()`, `ComplexChannel::getInputDims()`, `getLinearResponse()`, `ModelParam< double >::getVal()`, `Image< T >::getWidth()`, `NO_INIT`, `ComplexChannel::numChans()`, `numScales()`, `SQR()`, `subChan()`, `toPower()`, `uint`, and `Dims::w()`.

Referenced by `combineOutputs()`.

### 10.276.3.7 `void SoxChannel::killCaches ()` [protected, virtual]

Clear our cached inhibitory pool maps.

Reimplemented from [ComplexChannel](#).

Definition at line 258 of file SoxChannel.C.

References `ComplexChannel::killCaches()`.

### 10.276.3.8 `uint SoxChannel::numScales () const`

Get the number of scales used.

Definition at line 131 of file SoxChannel.C.

References `SingleChannel::getLevelSpec()`, `LevelSpec::maxDepth()`, `subChan()`, and `uint`.

Referenced by `combineOutputs()`, and `getNonlinearResponse()`.

### 10.276.3.9 `bool SoxChannel::setModelParamString (const std::string & name, const std::string & value, const bool recurse = false, const bool warn = true)` [protected, virtual]

Overload so that we can reconfigure when our params get changed.

Reimplemented from [ModelComponent](#).

Definition at line 91 of file SoxChannel.C.

References `buildSubChans()`, `ModelParamBase::getName()`, `ModelParam< uint >::getVal()`, `ComplexChannel::numChans()`, and `ModelComponent::setModelParamString()`.

### 10.276.3.10 `GaborChannel & SoxChannel::subChan (const uint idx) const` [inline, virtual]

Wraps `ComplexChannel`'s `subChan()` and casts the result to [GaborChannel](#).

Reimplemented from [ComplexChannel](#).

Definition at line 123 of file SoxChannel.C.

References [ComplexChannel::subChan\(\)](#), and [uint](#).

Referenced by [doInput\(\)](#), [getLinearResponse\(\)](#), [getNonlinearResponse\(\)](#), and [numScales\(\)](#).

## 10.276.4 Member Data Documentation

### 10.276.4.1 [ModelParam<uint>](#) [SoxChannel::itsNumOrients](#) [protected]

number of Gabor subchannels

Definition at line 76 of file SoxChannel.H.

The documentation for this class was generated from the following files:

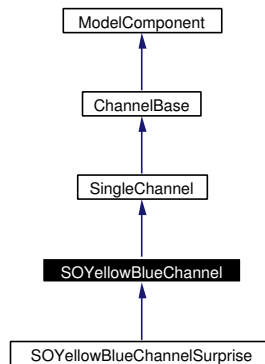
- [SoxChannel.H](#)
- [SoxChannel.C](#)

## 10.277 SOYellowBlueChannel Class Reference

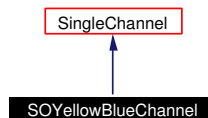
Single Opponent YellowBlueChannel member definitions:.

```
#include <Channels.H>
```

Inheritance diagram for SOYellowBlueChannel:



Collaboration diagram for SOYellowBlueChannel:



### Public Member Functions

- [SOYellowBlueChannel](#) ([ModelManager](#) &mgr)  
*Construct with standard params.*
- virtual [~SOYellowBlueChannel](#) ()  
*Destructor.*

### 10.277.1 Detailed Description

Single Opponent YellowBlueChannel member definitions:.

Definition at line 173 of file Channels.H.

## 10.277.2 Constructor & Destructor Documentation

### 10.277.2.1 SOYellowBlueChannel::SOYellowBlueChannel ([ModelManager](#) & *mgr*)

Construct with standard params.

Definition at line 217 of file Channels.C.

References [SOYB](#), and [SingleChannel::useSingleOpponence\(\)](#).

### 10.277.2.2 SOYellowBlueChannel::~~SOYellowBlueChannel () [virtual]

Destructor.

Definition at line 224 of file Channels.C.

The documentation for this class was generated from the following files:

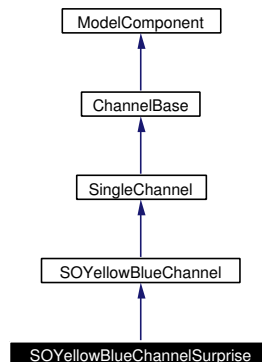
- [Channels.H](#)
- [Channels.C](#)

## 10.278 SOYellowBlueChannelSurprise Class Reference

Single Opponent YellowBlueChannelSurprise member definitions:.

```
#include <ChannelsSurprise.H>
```

Inheritance diagram for SOYellowBlueChannelSurprise:



Collaboration diagram for SOYellowBlueChannelSurprise:

### Public Member Functions

- [SOYellowBlueChannelSurprise](#) ([ModelManager](#) &mgr)  
*Construct with standard params.*
- virtual [~SOYellowBlueChannelSurprise](#) ()  
*Destructor.*
- virtual [Image](#)< float > [computeSubmap](#) (const [uint](#) index) const  
*[computeSubmap\(\)](#) is special and will return surprise*

### Protected Attributes

- [SharedPtr](#)< [SingleChannelSurprise](#) > itsSCS  
*our surprise computer*



### 10.278.1 Detailed Description

Single Opponent YellowBlueChannelSurprise member definitions:.

Definition at line 177 of file ChannelsSurprise.H.

### 10.278.2 Constructor & Destructor Documentation

#### 10.278.2.1 SOYellowBlueChannelSurprise::SOYellowBlueChannelSurprise (ModelManager & mgr)

Construct with standard params.

Definition at line 169 of file ChannelsSurprise.C.

References ModelComponent::addSubComponent().

#### 10.278.2.2 SOYellowBlueChannelSurprise::~~SOYellowBlueChannelSurprise () [virtual]

Destructor.

Definition at line 175 of file ChannelsSurprise.C.

### 10.278.3 Member Function Documentation

#### 10.278.3.1 Image< float > SOYellowBlueChannelSurprise::computeSubmap (const uint index) const [virtual]

computeSubmap() is special and will return surprise

Reimplemented from SingleChannel.

Definition at line 178 of file ChannelsSurprise.C.

References uint.

### 10.278.4 Member Data Documentation

#### 10.278.4.1 SharedPtr<SingleChannelSurprise> SOYellowBlueChannelSurprise::itsSCS [protected]

our surprise computer

Definition at line 190 of file ChannelsSurprise.H.

The documentation for this class was generated from the following files:

- [ChannelsSurprise.H](#)
- ChannelsSurprise.C

## 10.279 SpeedSensor Class Reference

This class allows to use a PS/2 mouse as a speed sensor.

```
#include <SpeedSensor.H>
```

### Public Member Functions

- void **go** ()
- float **getSpeed** ()
- int **getPos** () const
- void **setPos** (int)

#### 10.279.1 Detailed Description

This class allows to use a PS/2 mouse as a speed sensor.

it is multithreaded and will also give the position

Definition at line 45 of file SpeedSensor.H.

The documentation for this class was generated from the following files:

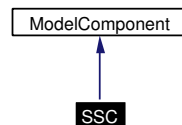
- [SpeedSensor.H](#)
- [SpeedSensor.C](#)

## 10.280 SSC Class Reference

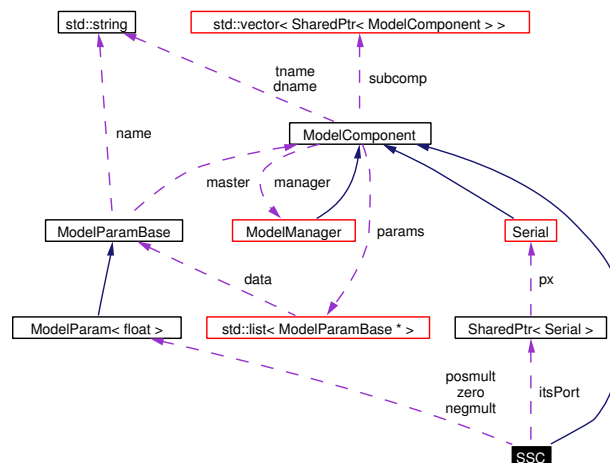
Interface to a serial-servo-controller.

```
#include <ssc.H>
```

Inheritance diagram for SSC:



Collaboration diagram for SSC:



### Public Member Functions

- `SSC` (`ModelManager` &mgr, const `std::string` &descrName="Mini-SSC Servo Driver", const `std::string` &tagName="MiniSSCdriver", const char \*defdev="/dev/ttyS0")

Default constructor; see [ModelComponent.H](#).

- `~SSC` ()

Destructor.

- bool `move` (const int servo, const float position)

*Moves servo # to given position in [-1.0 .. 1.0].*

- float `getPosition` (const int servo) const  
*Gets the current position of given servo.*
- void `calibrate` (const int servo, const byte neutralval, const byte minval, const byte maxval)  
*Calibrate a servo.*
- bool `moveRaw` (const int servo, const byte rawpos)  
*Moves servo # to given RAW (uncalibrated) position in [0..255].*
- bool `moveRawHack` (const int servo, const byte rawpos, const int port=1)  
*hack work around for moveRaw. Remove when serial stuff is fixed*
- byte `getPositionRaw` (const int servo) const  
*Gets the current RAW (uncalibrated) position [0..255] of given servo.*

## Protected Member Functions

- float `rawToCalib` (const int servo, const byte rawpos) const  
*Convert from raw (0..255) to calibrated (-1.0..1.0) position.*
- byte `calibToRaw` (const int servo, const float position) const  
*Convert from calibrated (-1.0..1.0) to raw (0..255) position.*

## Protected Attributes

- `SharedPtr< Serial > itsPort`  
*Serial port to use.*
- `ModelParam< float > * zero`  
*zero calibration values*
- `ModelParam< float > * posmult`  
*positive multiplier calib*
- `ModelParam< float > * negmult`  
*negative multiplier calib*

- `byte * pos`

*raw servo positions [0..255]*

### 10.280.1 Detailed Description

Interface to a serial-servo-controller.

The [ssc.C](#) and [ssc.H](#) files has the mini\_SSC controller code that help control the servos via the controller. The class is SSC class. Its functions include moving a servo to a specified position, using a serial port. There are two ways to move the servos and that their current positions: either using calibrated values in [-1.0..1.0], or using raw positions in [0..255]. Calibrated positions are converted to raw using two linear ramps (one for each side of the neutral position) that can be set using the [calibrate\(\)](#) function. EXAMPLE: if you do a `calibrate(servo, 100, 20, 150)`, then, subsequently, `move(servo, 0.0F)` will be equivalent to `moveRaw(servo, 100)`, `move(servo, -1.0F)` will be equivalent to `moveRaw(servo, 20)` and `move(servo, 1.0F)` will be equivalent to `moveRaw(servo, 150)`.

Definition at line 59 of file `ssc.H`.

### 10.280.2 Constructor & Destructor Documentation

**10.280.2.1** `SSC::SSC (ModelManager & mgr, const std::string & descrName = "Mini-SSC Servo Driver", const std::string & tagName = "MiniSSCdriver", const char * defdev = "/dev/ttyS0")`

Default constructor; see [ModelComponent.H](#).

Definition at line 42 of file `ssc.C`.

References `ModelComponent::addSubComponent()`, `ModelParam< float >::init()`, and `uint`.

**10.280.2.2** `SSC::~SSC ()`

Destructor.

Definition at line 68 of file `ssc.C`.

### 10.280.3 Member Function Documentation

#### 10.280.3.1 void SSC::calibrate (const int *servo*, const **byte** *neutralval*, const **byte** *minval*, const **byte** *maxval*)

Calibrate a servo.

Calibration will be made so that move(servo, 0.0F) will send the value 'neutralval' to the servo, move(servo, -1.0F) will send minval and move(servo, 1.0F) will send maxval.

Definition at line 84 of file ssc.C.

References ModelParam< float >::setVal().

#### 10.280.3.2 **byte** SSC::calibToRaw (const int *servo*, const float *position*) const [protected]

Convert from calibrated (-1.0..1.0) to raw (0..255) position.

Definition at line 173 of file ssc.C.

References byte.

Referenced by move().

#### 10.280.3.3 float SSC::getPosition (const int *servo*) const

Gets the current position of given servo.

Definition at line 76 of file ssc.C.

References rawToCalib().

#### 10.280.3.4 **byte** SSC::getPositionRaw (const int *servo*) const

Gets the current RAW (uncalibrated) position [0..255] of given servo.

Definition at line 144 of file ssc.C.

#### 10.280.3.5 bool SSC::move (const int *servo*, const float *position*)

Moves servo # to given position in [-1.0 .. 1.0].

Returns true on success, false if some serial error occurred.

Definition at line 72 of file ssc.C.

References calibToRaw(), and moveRaw().

**10.280.3.6** `bool SSC::moveRaw (const int servo, const byte rawpos)`

Moves servo # to given RAW (uncalibrated) position in [0..255].

Returns true on success, false if some serial error occurred.

Definition at line 96 of file ssc.C.

Referenced by `move()`.

**10.280.3.7** `bool SSC::moveRawHack (const int servo, const byte rawpos, const int port = 1)`

hack work around for moveRaw. Remove when serial stuff is fixed

Definition at line 119 of file ssc.C.

**10.280.3.8** `float SSC::rawToCalib (const int servo, const byte rawpos) const [protected]`

Convert from raw (0..255) to calibrated (-1.0..1.0) position.

Definition at line 153 of file ssc.C.

References `byte`, and `ModelParam< float >::getVal()`.

Referenced by `getPosition()`.

**10.280.4 Member Data Documentation****10.280.4.1** `SharedPtr<Serial> SSC::itsPort [protected]`

[Serial](#) port to use.

Definition at line 96 of file ssc.H.

**10.280.4.2** `ModelParam<float>* SSC::negmult [protected]`

negative multiplier calib

Definition at line 99 of file ssc.H.

**10.280.4.3** `byte* SSC::pos [protected]`

raw servo positions [0..255]

Definition at line 100 of file ssc.H.



**10.280.4.4** [ModelParam<float>\\*](#) [SSC::posmult](#) [protected]

positive multiplier calib

Definition at line 98 of file ssc.H.

**10.280.4.5** [ModelParam<float>\\*](#) [SSC::zero](#) [protected]

zero calibration values

Definition at line 97 of file ssc.H.

The documentation for this class was generated from the following files:

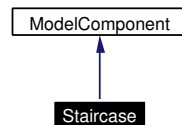
- [ssc.H](#)
- [ssc.C](#)

## 10.281 Staircase Class Reference

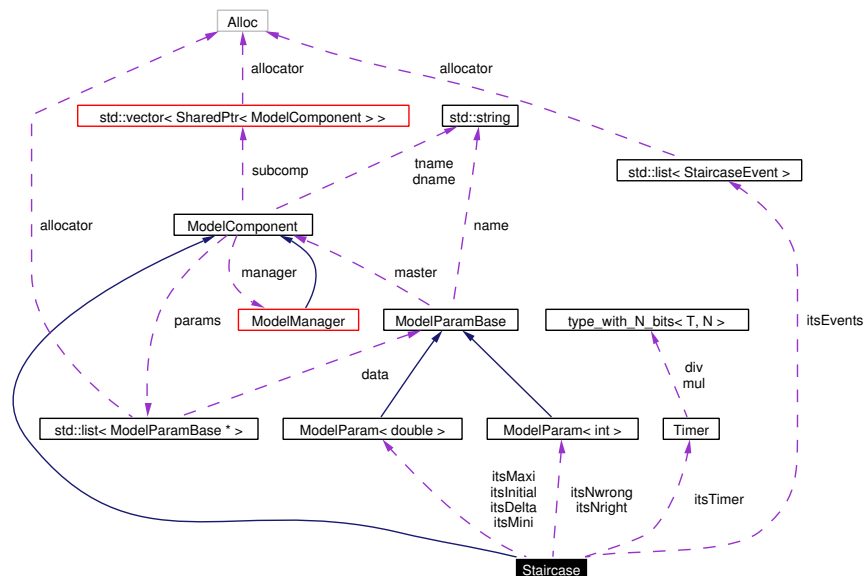
A staircase procedure to measure psychophysical thresholds.

```
#include <Staircase.H>
```

Inheritance diagram for Staircase:



Collaboration diagram for Staircase:



### Public Member Functions

- **Staircase** (**ModelManager** &mgr, const **std::string** &descrName="Staircase", const **std::string** &tagName="Staircase")

*Constructor.*

- **~Staircase** ()

*Destructor.*

- void [reset](#) (const bool recurse=true)  
*Reset to initial conditions.*
- void [getValues](#) (double &value1, double &value2)  
*Get parameter values to use for next trial.*
- void [setResponse](#) (const bool response)  
*Set the response of the current trial.*
- void [pushEvent](#) (const [std::string](#) &msg, const int trial=-1, const double val=0.0, const bool response=false)  
*Log an event.*

## Protected Member Functions

- void [start2](#) ()  
*get started*
- void [stop1](#) ()  
*get stopped*

## Protected Attributes

- [ModelParam](#)< double > [itsInitial](#)  
*initial value*
- [ModelParam](#)< double > [itsDelta](#)  
*delta in values*
- [ModelParam](#)< double > [itsMini](#)  
*minimum possible value*
- [ModelParam](#)< double > [itsMaxi](#)  
*maximum possible value*
- [ModelParam](#)< int > [itsNright](#)  
*nb of right responses to subtract a delta*

- [ModelParam](#)< int > [itsNwrong](#)  
*nb of wrong responses to add a delta*
- [ModelParam](#)< std::string > [itsFileName](#)  
*name of file to save the logs*

### 10.281.1 Detailed Description

A staircase procedure to measure psychophysical thresholds.

This class facilitates the evaluation of a threshold in a psychophysical task.

Definition at line 49 of file Staircase.H.

### 10.281.2 Constructor & Destructor Documentation

#### 10.281.2.1 [Staircase::Staircase](#) ([ModelManager](#) & *mgr*, const [std::string](#) & *descrName* = "Staircase", const [std::string](#) & *tagName* = "Staircase")

Constructor.

Definition at line 43 of file Staircase.C.

#### 10.281.2.2 [Staircase::~~Staircase](#) ()

Destructor.

Definition at line 59 of file Staircase.C.

### 10.281.3 Member Function Documentation

#### 10.281.3.1 [void Staircase::getValues](#) (double & *value1*, double & *value2*)

Get parameter values to use for next trial.

This will give you two values, one to use for the first period of the following trial, and one for the second period. The Staircase class will keep in memory what the expected answer should be, which you will give using [setResponse\(\)](#).

Definition at line 124 of file Staircase.C.

References [pushEvent\(\)](#).

**10.281.3.2 void Staircase::pushEvent (const [std::string](#) & msg, const int trial = -1, const double val = 0.0, const bool response = false)**

Log an event.

This is in the public section just in case someone wants to push external events, but should not be called by the user in normal situations, as it is called internally when getNextValue() is called.

Definition at line 167 of file Staircase.C.

References [Timer::get\(\)](#).

Referenced by [getValues\(\)](#), [reset\(\)](#), [setResponse\(\)](#), and [start2\(\)](#).

**10.281.3.3 void Staircase::reset (const bool recurse = true) [virtual]**

Reset to initial conditions.

Reimplemented from [ModelComponent](#).

Definition at line 75 of file Staircase.C.

References [ModelParam< double >::getVal\(\)](#), [itsInitial](#), [pushEvent\(\)](#), [ModelComponent::reset\(\)](#), and [Timer::reset\(\)](#).

**10.281.3.4 void Staircase::setResponse (const bool response)**

Set the response of the current trial.

The response should be true is the user said nothing-then-something, false otherwise.

Definition at line 143 of file Staircase.C.

References [ModelParam< double >::getVal\(\)](#), [ModelParam< int >::getVal\(\)](#), [itsDelta](#), [itsMaxi](#), [itsMini](#), [itsNright](#), [itsNwrong](#), and [pushEvent\(\)](#).

**10.281.3.5 void Staircase::start2 () [protected, virtual]**

get started

Reimplemented from [ModelComponent](#).

Definition at line 63 of file Staircase.C.

References [ModelParam< double >::getVal\(\)](#), [itsInitial](#), [pushEvent\(\)](#), and [Timer::reset\(\)](#).

**10.281.3.6 void Staircase::stop1 ()** [protected, virtual]

get stopped

Reimplemented from [ModelComponent](#).

Definition at line 87 of file Staircase.C.

**10.281.4 Member Data Documentation****10.281.4.1 [ModelParam](#)<double> [Staircase::itsDelta](#)** [protected]

delta in values

Definition at line 85 of file Staircase.H.

Referenced by [setResponse\(\)](#).

**10.281.4.2 [ModelParam](#)<std::string> [Staircase::itsFileName](#)** [protected]

name of file to save the logs

Definition at line 90 of file Staircase.H.

**10.281.4.3 [ModelParam](#)<double> [Staircase::itsInitial](#)** [protected]

initial value

Definition at line 84 of file Staircase.H.

Referenced by [reset\(\)](#), and [start2\(\)](#).

**10.281.4.4 [ModelParam](#)<double> [Staircase::itsMaxi](#)** [protected]

maximum possible value

Definition at line 87 of file Staircase.H.

Referenced by [setResponse\(\)](#).

**10.281.4.5 [ModelParam](#)<double> [Staircase::itsMini](#)** [protected]

minimum possible value

Definition at line 86 of file Staircase.H.

Referenced by [setResponse\(\)](#).

**10.281.4.6** [ModelParam<int> Staircase::itsNright](#) [protected]

nb of right responses to subtract a delta

Definition at line 88 of file Staircase.H.

Referenced by [setResponse\(\)](#).

**10.281.4.7** [ModelParam<int> Staircase::itsNwrong](#) [protected]

nb of wrong responses to add a delta

Definition at line 89 of file Staircase.H.

Referenced by [setResponse\(\)](#).

The documentation for this class was generated from the following files:

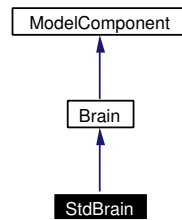
- [Staircase.H](#)
- [Staircase.C](#)

## 10.282 StdBrain Class Reference

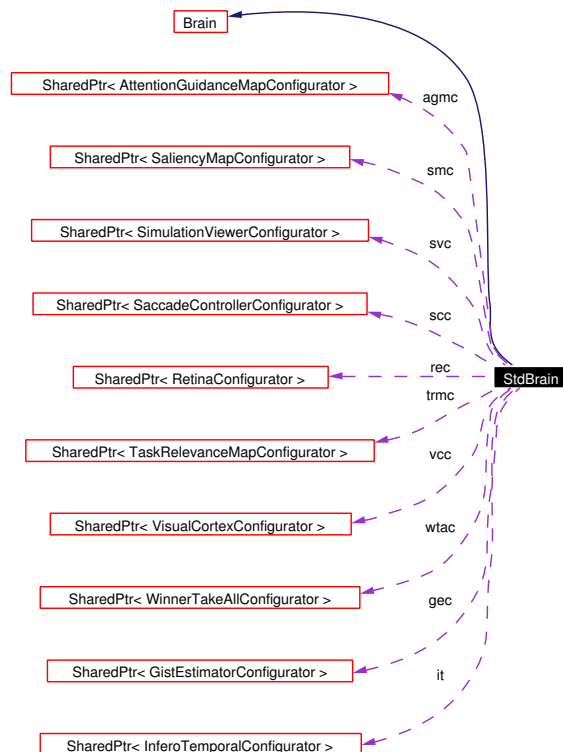
A subclass of [Brain](#) that automatically sets up the standard modules.

```
#include <StdBrain.H>
```

Inheritance diagram for StdBrain:



Collaboration diagram for StdBrain:





## Public Member Functions

- [StdBrain](#) ([ModelManager](#) &mgr, const [std::string](#) &descrName="Brain", const [std::string](#) &tagName="Brain")  
*Construct using a [VisualCortex](#) object with standard channels.*
- virtual [~StdBrain](#) ()  
*virtual destructor for safe inheritance*

## Protected Member Functions

- virtual void [start1](#) ()  
*get started (in particular, instantiate [SaccadeController](#))*

### 10.282.1 Detailed Description

A subclass of [Brain](#) that automatically sets up the standard modules.

StdBrain contains a [VisualCortex](#), [ShapeEstimator](#), [SaccadeController](#), [SaliencyMap](#), [TaskRelevanceMap](#), [AttentionGuidanceMap](#), [WinnerTakeAll](#) and [SimulationViewer](#) as "modules" (see [Brain.H](#)). Module types to be used can be configured via the command-line thanks to the use of the appropriate Configurators here.

Definition at line 60 of file StdBrain.H.

### 10.282.2 Constructor & Destructor Documentation

#### 10.282.2.1 StdBrain::StdBrain ([ModelManager](#) & mgr, const [std::string](#) & descrName = "Brain", const [std::string](#) & tagName = "Brain")

Construct using a [VisualCortex](#) object with standard channels.

##### Parameters:

- mgr* our [ModelManager](#) (see [ModelManager.H](#))
- descrName* descriptive name for human useage
- tagName* name for [ParamMap](#) usage

Definition at line 56 of file StdBrain.C.

References [Brain::addModule\(\)](#), and [ModelComponent::addSubComponent\(\)](#).

### 10.282.2.2 StdBrain::~~StdBrain () [virtual]

virtual destructor for safe inheritance

Definition at line 92 of file StdBrain.C.

## 10.282.3 Member Function Documentation

### 10.282.3.1 void StdBrain::start1 () [protected, virtual]

get started (in particular, instantiate [SaccadeController](#))

Reimplemented from [Brain](#).

Definition at line 96 of file StdBrain.C.

References [Brain::addModule\(\)](#), [SharedPtr< GistEstimatorConfigurator >::get\(\)](#), [SharedPtr< IneroTemporalConfigurator >::get\(\)](#), [SharedPtr< SimulationViewerConfigurator >::get\(\)](#), [SharedPtr< WinnerTakeAllConfigurator >::get\(\)](#), [SharedPtr< AttentionGuidanceMapConfigurator >::get\(\)](#), [SharedPtr< TaskRelevanceMapConfigurator >::get\(\)](#), [SharedPtr< SaliencyMapConfigurator >::get\(\)](#), [SharedPtr< VisualCortexConfigurator >::get\(\)](#), [SharedPtr< SaccadeControllerConfigurator >::get\(\)](#), [SharedPtr< RetinaConfigurator >::get\(\)](#), [Brain::getGE\(\)](#), [Brain::getRET\(\)](#), [Brain::getSC\(\)](#), [Brain::getSE\(\)](#), [Brain::getSV\(\)](#), [Brain::getTRM\(\)](#), [Brain::getVC\(\)](#), [Brain::hasModule\(\)](#), and [Brain::start1\(\)](#).

The documentation for this class was generated from the following files:

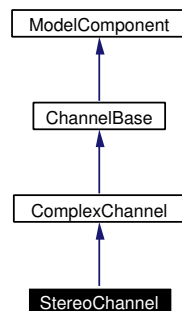
- [StdBrain.H](#)
- [StdBrain.C](#)

## 10.283 StereoChannel Class Reference

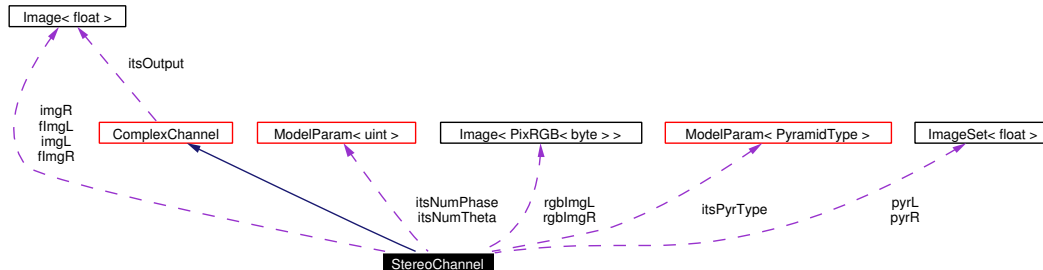
A composite channel containing a set of Disparity channels.

```
#include <Channels.H>
```

Inheritance diagram for StereoChannel:



Collaboration diagram for StereoChannel:



### Public Member Functions

- [StereoChannel](#) ([ModelManager](#) &[mgr](#))  
*Construct.*
- virtual [~StereoChannel](#) ()  
*destructor*
- virtual void [exportOptions](#) (const int whatToExport, const bool recurse=true)  
*Export some of our [ModelParam](#) parameters as command-line-options.*

- virtual [DisparityChannel](#) & [dispChan](#) (const [uint](#) idO, const [uint](#) idP) const  
*Wraps [ComplexChannel](#)'s [subChan\(\)](#) and casts the result to [DisparityChannel](#).*
- virtual bool [setModelParamString](#) (const [std::string](#) &name, const [std::string](#) &value, const bool recurse=false, const bool warn=true)  
*Overload so that we can reconfigure when our params get changed.*
- void [setSecondImage](#) (const [Image](#)< float > \*bwimg)  
*set the second image to complete stereo data*
- void [getRawFilteredImages](#) ([Image](#)< float > \*\*\*\*fImgLE, [Image](#)< float > \*\*\*\*fImgRI)  
*access method for easy viewing*

## Protected Member Functions

- virtual void [buildSubChans](#) ()  
*(re-)build our subchannels*
- virtual void [doInput](#) (const [Image](#)< [PixRGB](#)< float > > \*, const [Image](#)< float > \*bwimg, const double t, const [Image](#)< [byte](#) > &clipMask)  
*StereoChannel only required luminosity input AT THIS POINT.*
- virtual void [start1](#) ()  
*overload [start1\(\)](#)*

## Protected Attributes

- [ModelParam](#)< [PyramidType](#) > [itsPyrType](#)  
*type of pyramid to use in our [DisparityChannel](#) subcomponents*
- [ModelParam](#)< [uint](#) > [itsNumTheta](#)  
*number of orientations of disparity subchannels*
- [ModelParam](#)< [uint](#) > [itsNumPhase](#)  
*number of phases of each orientations of disparity subchannels*
- int [itsExportOpts](#)  
*keep track of what options to export in case we rebuild our subchans*

### 10.283.1 Detailed Description

A composite channel containing a set of Disparity channels.

Definition at line 1189 of file Channels.H.

### 10.283.2 Constructor & Destructor Documentation

#### 10.283.2.1 StereoChannel::StereoChannel ([ModelManager](#) & *mgr*)

Construct.

Definition at line 2231 of file Channels.C.

References [buildSubChans\(\)](#), [Gaussian3](#), [OPTEXP\\_NONE](#), and [STEREO](#).

#### 10.283.2.2 StereoChannel::~~StereoChannel () [virtual]

destructor

Definition at line 2247 of file Channels.C.

### 10.283.3 Member Function Documentation

#### 10.283.3.1 void StereoChannel::buildSubChans () [protected, virtual]

(re-)build our subchannels

Definition at line 2286 of file Channels.C.

References [ComplexChannel::addSubChan\(\)](#), [dispChan\(\)](#), [ModelComponent::exportOptions\(\)](#), [ModelParam< uint >::getVal\(\)](#), [itsNumPhase](#), [itsNumTheta](#), [makeShared\(\)](#), [OPTEXP\\_NONE](#), [ModelComponent::removeAllSubComponents\(\)](#), [ChannelBase::setHomogeneous\(\)](#), [DisparityChannel::setNumPhase\(\)](#), [DisparityChannel::setNumTheta\(\)](#), and [uint](#).

Referenced by [setModelParamString\(\)](#), and [StereoChannel\(\)](#).

#### 10.283.3.2 [DisparityChannel](#) & StereoChannel::dispChan (const [uint](#) *idO*, const [uint](#) *idP*) const [virtual]

Wraps [ComplexChannel](#)'s [subChan\(\)](#) and casts the result to [DisparityChannel](#).

Definition at line 2278 of file Channels.C.

References `ModelParam< uint >::getVal()`, `itsNumPhase`, `ComplexChannel::subChan()`, and `uint`.

Referenced by `buildSubChans()`, `doInput()`, and `start1()`.

**10.283.3.3** `void StereoChannel::doInput (const Image< PixRGB< float > > *, const Image< float > * bwimg, const double t, const Image< byte > & clipMask)` [protected, virtual]

StereoChannel only required luminosity input AT THIS POINT.

Implements [ChannelBase](#).

Definition at line 2399 of file `Channels.C`.

References `buildPyrGaussian()`, `dispChan()`, `DisparityChannel::doInput()`, `ModelParam< uint >::getVal()`, `itsNumPhase`, `itsNumTheta`, `DisparityChannel::setRawFilteredImages()`, and `uint`.

**10.283.3.4** `void StereoChannel::exportOptions (const int whatToExport, const bool recurse = true)` [virtual]

Export some of our [ModelParam](#) parameters as command-line-options.

See the base function in [ModelComponent.H](#) for info.

Reimplemented from [ComplexChannel](#).

Definition at line 2262 of file `Channels.C`.

References `ComplexChannel::exportOptions()`, `itsNumTheta`, `OPTEXP_CORE`, and `ModelManager::requestOption()`.

**10.283.3.5** `void StereoChannel::getRawFilteredImages (Image< float > **** fImgLE, Image< float > **** fImgRI)`

access method for easy viewing

Definition at line 2462 of file `Channels.C`.

**10.283.3.6** `bool StereoChannel::setModelParamString (const std::string & name, const std::string & value, const bool recurse = false, const bool warn = true)` [virtual]

Overload so that we can reconfigure when our params get changed.

Reimplemented from [ModelComponent](#).

Definition at line 2378 of file `Channels.C`.

References [buildSubChans\(\)](#), [ModelParamBase::getName\(\)](#), [ModelParam< uint >::getVal\(\)](#), [itsNumPhase](#), [itsNumTheta](#), [ComplexChannel::numChans\(\)](#), and [ModelComponent::setModelParamString\(\)](#).

#### 10.283.3.7 void StereoChannel::setSecondImage (const [Image< float > \\*](#) *bwimg*)

set the second image to complete stereo data

Definition at line 2456 of file Channels.C.

#### 10.283.3.8 void StereoChannel::start1 () [protected, virtual]

overload [start1\(\)](#)

Reimplemented from [ModelComponent](#).

Definition at line 2251 of file Channels.C.

References [dispChan\(\)](#), [SingleChannel::getLevelSpec\(\)](#), and [LevelSpec::maxDepth\(\)](#).

### 10.283.4 Member Data Documentation

#### 10.283.4.1 int StereoChannel::itsExportOpts [protected]

keep track of what options to export in case we rebuild our subchans

Definition at line 1238 of file Channels.H.

#### 10.283.4.2 [ModelParam<uint>](#) StereoChannel::itsNumPhase [protected]

number of phases of each orientations of disparity subchannels

Definition at line 1227 of file Channels.H.

Referenced by [buildSubChans\(\)](#), [dispChan\(\)](#), [doInput\(\)](#), and [setModelParamString\(\)](#).

#### 10.283.4.3 [ModelParam<uint>](#) StereoChannel::itsNumTheta [protected]

number of orientations of disparity subchannels

Definition at line 1224 of file Channels.H.

Referenced by [buildSubChans\(\)](#), [doInput\(\)](#), [exportOptions\(\)](#), and [setModelParamString\(\)](#).

#### 10.283.4.4 [ModelParam<PyramidType> StereoChannel::itsPyrType](#) [protected]

type of pyramid to use in our [DisparityChannel](#) subcomponents

Definition at line 1221 of file Channels.H.

The documentation for this class was generated from the following files:

- [Channels.H](#)
- [Channels.C](#)

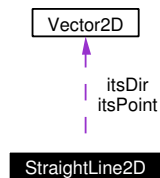


## 10.284 StraightLine2D Class Reference

A straight line in the 2D plane.

```
#include <Geometry2D.H>
```

Collaboration diagram for StraightLine2D:



### Public Member Functions

- [StraightLine2D](#) ()  
*default constructo, constructs an invalid object*
- [StraightLine2D](#) (const [Vector2D](#) &point, const [Vector2D](#) &direction)  
*construct from two [Vector2D](#)*
- [StraightLine2D](#) (std::istream &is)  
*Construct from the data contained in the input stream is.*
- void [reset](#) (const [Vector2D](#) &point, const [Vector2D](#) &direction)  
*reset the straight line*
- void [writeToStream](#) (std::ostream &os) const  
*write the entire StraightLine2D to the output stream os*
- void [readFromStream](#) (std::istream &is)  
*read the StraightLine2D from the input stream is*
- [Vector2D](#) [point](#) (float n=0.0F) const  
*get a point on the line*
- [Vector2D](#) [direction](#) () const  
*returns the direction vector for this straight line*
- [Vector2D](#) [intersect](#) (const [StraightLine2D](#) &other, float &n, float &m) const

*returns a vector to the intersection point of this and other*

- bool **isParallel** (const **StraightLine2D** &other) const  
*whether this and other are parallel*
- bool **isOrthogonal** (const **StraightLine2D** &other) const  
*whether this and other are orthogonal*
- bool **isPointOnLine** (const **Vector2D** &pt) const  
*whether pt is on the straight line*
- bool **isIdentical** (const **StraightLine2D** &other) const  
*whether this and other describe identical straight lines*
- bool **isValid** () const  
*whether this object is valid*

### 10.284.1 Detailed Description

A straight line in the 2D plane.

Definition at line 156 of file Geometry2D.H.

### 10.284.2 Constructor & Destructor Documentation

#### 10.284.2.1 **StraightLine2D::StraightLine2D ()**

default constructo, constructs an invalid object

Definition at line 298 of file Geometry2D.C.

#### 10.284.2.2 **StraightLine2D::StraightLine2D (const **Vector2D** & point, const **Vector2D** & direction)**

construct from two **Vector2D**

##### **Parameters:**

**point** vector determining a point on the line

**direction** vector determining the direction of the line

Definition at line 303 of file Geometry2D.C.

### 10.284.2.3 StraightLine2D::StraightLine2D (std::istream & is)

Construct from the data contained in the input stream is.

Definition at line 309 of file Geometry2D.C.

References readFromStream().

## 10.284.3 Member Function Documentation

### 10.284.3.1 [Vector2D](#) StraightLine2D::direction () const

returns the direction vector for this straight line

Definition at line 348 of file Geometry2D.C.

References ASSERT, and isValid().

Referenced by intersect(), isOrthogonal(), and isParallel().

### 10.284.3.2 [Vector2D](#) StraightLine2D::intersect (const [StraightLine2D](#) & other, float & n, float & m) const

returns a vector to the intersection point of this and other

if the lines do not intersect (i.e. they are parallel), an invalid [Vector2D](#) is returned.

#### Parameters:

**n** in this variable, the scalar is returned, for which the intersection point is this->point(n);

**m** the scalar, for which the intersection point is other.point(m)

Definition at line 355 of file Geometry2D.C.

References ASSERT, direction(), isParallel(), isValid(), point(), Vector2D::x(), and Vector2D::y().

### 10.284.3.3 bool StraightLine2D::isIdentical (const [StraightLine2D](#) & other) const

whether this and other describe identical straight lines

Definition at line 400 of file Geometry2D.C.

References ASSERT, isParallel(), isPointOnLine(), isValid(), and point().

**10.284.3.4 bool StraightLine2D::isOrthogonal (const [StraightLine2D](#) & other) const**

whether this and other are orthogonal

Definition at line 380 of file Geometry2D.C.

References ASSERT, direction(), Vector2D::isOrthogonal(), and isValid().

**10.284.3.5 bool StraightLine2D::isParallel (const [StraightLine2D](#) & other) const**

whether this and other are parallel

Definition at line 372 of file Geometry2D.C.

References ASSERT, direction(), Vector2D::isCollinear(), and isValid().

Referenced by intersect(), and isIdentical().

**10.284.3.6 bool StraightLine2D::isPointOnLine (const [Vector2D](#) & pt) const**

whether pt is on the straight line

Definition at line 388 of file Geometry2D.C.

References ASSERT, Vector2D::isValid(), isValid(), Vector2D::x(), and Vector2D::y().

Referenced by isIdentical().

**10.284.3.7 bool StraightLine2D::isValid () const**

whether this object is valid

Definition at line 409 of file Geometry2D.C.

Referenced by direction(), intersect(), isIdentical(), isOrthogonal(), isParallel(), isPointOnLine(), and point().

**10.284.3.8 [Vector2D](#) StraightLine2D::point (float *n* = 0.0F) const**

get a point on the line

**Parameters:**

*n* get point point + *n* \* direction; for *n* = 0 (default), this is the point entered in the constructor or in reset

Definition at line 341 of file Geometry2D.C.

References ASSERT, and isValid().

Referenced by intersect(), and isIdentical().

#### 10.284.3.9 void StraightLine2D::readFromStream (std::istream & is)

read the StraightLine2D from the input stream is

Definition at line 332 of file Geometry2D.C.

References Vector2D::readFromStream().

Referenced by Token::readFromStream(), and StraightLine2D().

#### 10.284.3.10 void StraightLine2D::reset (const [Vector2D](#) & point, const [Vector2D](#) & direction)

reset the straight line

##### Parameters:

*point* vector determining a point on the line

*direction* vector determining the direction of the line

Definition at line 315 of file Geometry2D.C.

#### 10.284.3.11 void StraightLine2D::writeToStream (std::ostream & os) const

write the entire StraightLine2D to the output stream os

Definition at line 323 of file Geometry2D.C.

References Vector2D::writeToStream().

Referenced by Token::writeToStream().

The documentation for this class was generated from the following files:

- [Geometry2D.H](#)
- [Geometry2D.C](#)

## 10.285 `std::string` Class Reference

The STL string class. Use with `#include` .

`#include <dummySTL.H>`

### 10.285.1 Detailed Description

The STL string class. Use with `#include` .

`std::string` is actually `std::basic_string<char>`. See [http://www.sgi.com/tech/stl/basic\\_string.html](http://www.sgi.com/tech/stl/basic_string.html) SGI's documentation of `std::basic_string`.

Definition at line 63 of file `dummySTL.H`.

The documentation for this class was generated from the following file:

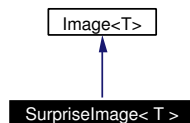
- [dummySTL.H](#)

## 10.286 SurpriseImage< T > Class Template Reference

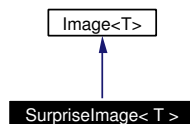
A 2D array of [SurpriseModel](#) objects.

```
#include <SurpriseImage.H>
```

Inheritance diagram for SurpriseImage< T >:



Collaboration diagram for SurpriseImage< T >:



### Public Member Functions

- [SurpriseImage](#) ()  
*Uninitialized constructor.*
- [SurpriseImage](#) (const double updfac, const [Image](#)< double > &sampleval, const [Image](#)< double > &samplevar)  
*Constructor from given sample means and variances.*
- [~SurpriseImage](#) ()  
*Virtual destructor ensures proper destruction of derived classes.*
- void [reset](#) ()  
*Reset all models to initial state.*
- void [init](#) (const double updfac, const [Image](#)< double > &sampleval, const [Image](#)< double > &samplevar)  
*Init to given sample mean and variance.*
- void [resetUpdFac](#) (const double updfac)

*reset our update factor*

- [Image](#)< double > [surprise](#) (const [SurpriseImage](#)< T > &other)

*Compute surprise between us and another model.*

- void [neighborhoods](#) (const [SurpriseImage](#)< T > &models, const float neigh-sigma)

*Initialize us as a weighted combination of the given map of models.*

- void [neighborhoods](#) (const [SurpriseImage](#)< T > &models, const [Image](#)< float > &weights, const float wmin)

*Initialize us as a weighted combination of the given map of models.*

- [Image](#)< double > [getMean](#) () const

*get our means*

- [Image](#)< double > [getVar](#) () const

*get our variances*

### 10.286.1 Detailed Description

**template<class T> class [SurpriseImage](#)< T >**

A 2D array of [SurpriseModel](#) objects.

This class is derived from [Image](#) and inherits all its functionality, and in addition provides batch access to several of the [SurpriseModel](#) functions. Although the template argument T could in principle be anything, it is required that it supports the [SurpriseModel](#) functions like [update\(\)](#), [surprise\(\)](#), etc

Definition at line 51 of file [SurpriseImage.H](#).

### 10.286.2 Constructor & Destructor Documentation

**10.286.2.1 template<class T> [SurpriseImage](#)< T >::[SurpriseImage](#) ()**

Uninitialized constructor.

Definition at line 42 of file [SurpriseImage.C](#).



**10.286.2.2** `template<class T> SurpriseImage< T >::SurpriseImage (const double updfac, const Image< double > & sampleval, const Image< double > & samplevar)`

Constructor from given sample means and variances.

Definition at line 48 of file `SurpriseImage.C`.

References `SurpriseImage< T >::init()`, and `NO_INIT`.

**10.286.2.3** `template<class T> SurpriseImage< T >::~~SurpriseImage ()`

Virtual destructor ensures proper destruction of derived classes.

Definition at line 56 of file `SurpriseImage.C`.

## 10.286.3 Member Function Documentation

**10.286.3.1** `template<class T> Image< double > SurpriseImage< T >::getMean () const`

get our means

Reimplemented from `Image< T >`.

Definition at line 145 of file `SurpriseImage.C`.

References `Image< T >::begin()`, `Image< T >::beginw()`, `Image< T >::end()`, `Image< T >::getMean()`, and `NO_INIT`.

Referenced by `SurpriseMap< T >::surprise()`.

**10.286.3.2** `template<class T> Image< double > SurpriseImage< T >::getVar () const`

get our variances

Definition at line 159 of file `SurpriseImage.C`.

References `Image< T >::begin()`, `Image< T >::beginw()`, `Image< T >::end()`, and `NO_INIT`.

Referenced by `SurpriseMap< T >::surprise()`.

**10.286.3.3** `template<class T> void SurpriseImage< T >::init (const double updfac, const Image< double > & sampleval, const Image< double > & samplevar)`

Init to given sample mean and variance.

Definition at line 69 of file `SurpriseImage.C`.

References `ASSERT`, `Image< T >::begin()`, `Image< T >::beginw()`, `Image< T >::endw()`, and `Image< T >::isSameSize()`.

Referenced by `SurpriseImage< T >::SurpriseImage()`.

**10.286.3.4** `template<class T> void SurpriseImage< T >::neighborhoods (const SurpriseImage< T > & models, const Image< float > & weights, const float wmin)`

Initialize us as a weighted combination of the given map of models.

This will initialize us so that each of our pixels is built from the [SurpriseModel](#) `combineFrom()` function. In this version, weights should be an image with double+1 the width and height of models.

Definition at line 127 of file `SurpriseImage.C`.

References `ASSERT`, `Image< T >::beginw()`, `Image< T >::getHeight()`, `Image< T >::getWidth()`, `Point2D::i`, and `Point2D::j`.

**10.286.3.5** `template<class T> void SurpriseImage< T >::neighborhoods (const SurpriseImage< T > & models, const float neighsigma)`

Initialize us as a weighted combination of the given map of models.

This will initialize us so that each of our pixels is built from the [SurpriseModel](#) `combineFrom()` function, with a Gaussian weighing mask centered onto that pixel and with sigma given here as argument.

Definition at line 108 of file `SurpriseImage.C`.

References `Image< T >::beginw()`, `Image< T >::getDims()`, `Image< T >::getHeight()`, `Image< T >::getWidth()`, `Point2D::i`, and `Point2D::j`.

Referenced by `SurpriseMap< T >::surprise()`.

**10.286.3.6** `template<class T> void SurpriseImage< T >::reset ()`

Reset all models to initial state.

Definition at line 61 of file `SurpriseImage.C`.

References Image< T >::beginw(), and Image< T >::endw().

Referenced by SurpriseMap< T >::surprise().

#### 10.286.3.7 `template<class T> void SurpriseImage< T >::resetUpdFac (const double updfac)`

reset our update factor

Definition at line 83 of file SurpriseImage.C.

References Image< T >::beginw(), and Image< T >::endw().

Referenced by SurpriseMap< T >::surprise().

#### 10.286.3.8 `template<class T> Image< double > SurpriseImage< T >::surprise (const SurpriseImage< T > & other)`

Compute surprise between us and another model.

Definition at line 91 of file SurpriseImage.C.

References ASSERT, Image< T >::begin(), Image< T >::beginw(), Image< T >::endw(), and NO\_INIT.

Referenced by SurpriseMap< T >::surprise().

The documentation for this class was generated from the following files:

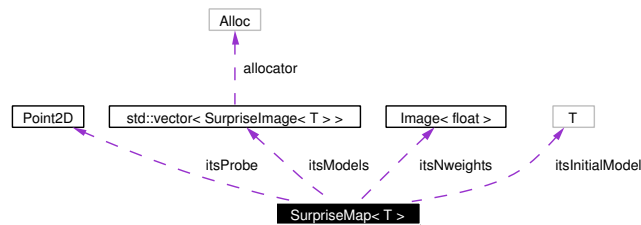
- [SurpriseImage.H](#)
- [SurpriseImage.C](#)

## 10.287 SurpriseMap< T > Class Template Reference

A Surprise [Map](#).

```
#include <SurpriseMap.H>
```

Collaboration diagram for SurpriseMap< T >:



### Public Member Functions

- [SurpriseMap \(\)](#)

*Constructor; you must call [init\(\)](#) before using.*

- void [init](#) (const [uint](#) qlen=1, const double updatefac=0.9, const double neighupdatefac=0.5, const double sampleval=0.0, const double samplevar=1.0, const float neighsigma=0.5f, const float locsigma=3.0f, const [Point2D](#) &probe=[Point2D](#)(-1,-1))

*initialization*

- [~SurpriseMap \(\)](#)

*Virtual destructor ensures proper destruction of derived classes.*

- void [reset](#) ()

*Reset to initial state.*

- [Image< double > surprise](#) (const [SurpriseImage< T >](#) &sample)

*Update by receiving a new sample and sample measurement error.*

- const [SurpriseImage< T >](#) & [getSurpriseImage](#) (const [uint](#) index) const

*Access to underlying [SurpriseImage](#) members:.*

### 10.287.1 Detailed Description

**template<class T> class SurpriseMap< T >**

A Surprise [Map](#).

This class contains a [SurpriseImage](#) of our current models at each pixel, plus an [Image-Cache](#) of [SurpriseImage](#) objects built to represent the neighborhood distributions.

Definition at line 52 of file SurpriseMap.H.

### 10.287.2 Constructor & Destructor Documentation

**10.287.2.1    template<class T> [SurpriseMap](#)< T >::[SurpriseMap](#) ()**

Constructor; you must call [init\(\)](#) before using.

Definition at line 43 of file SurpriseMap.C.

**10.287.2.2    template<class T> [SurpriseMap](#)< T >::~~[SurpriseMap](#) ()**

Virtual destructor ensures proper destruction of derived classes.

Definition at line 68 of file SurpriseMap.C.

### 10.287.3 Member Function Documentation

**10.287.3.1    template<class T> const [SurpriseImage](#)< T > & [SurpriseMap](#)< T >::[getSurpriseImage](#) (const [uint](#) *index*) const**

Access to underlying [SurpriseImage](#) members:.

Definition at line 184 of file SurpriseMap.C.

References [ASSERT](#), and [uint](#).

**10.287.3.2    template<class T> void [SurpriseMap](#)< T >::[init](#) (const [uint](#) *qlen* = 1, const double *updatefac* = 0.9, const double *neighupdatefac* = 0.5, const double *sampleval* = 0.0, const double *samplevar* = 1.0, const float *neighsigma* = 0.5f, const float *locsigma* = 3.0f, const [Point2D](#) & *probe* = [Point2D](#)(-1,-1))**

initialization

**Parameters:**

*qlen*    number of maps

**updatefac** factor by which the current model is combined with a new model on an update. Valid values are in [0..1]

**neighupdatefac** factor by which local data is combined with neighborhood models, or 0.0 to use the value of updatefac

**sampleval** the value of the samples previously received.

**samplevar** a variance expressing the intrinsic variability in the measurement of those samples.

**neighsigma** sigma to use for the neighborhoods as a fraction of max(map width, map height)

**locsigma** sigma to use for local neighborhood, in pixels

Definition at line 51 of file SurpriseMap.C.

References Image< float >::freeMem(), and uint.

Referenced by SingleChannelSurprise::start1().

### 10.287.3.3 `template<class T> void SurpriseMap< T >::reset ()`

Reset to initial state.

Definition at line 73 of file SurpriseMap.C.

References uint.

### 10.287.3.4 `template<class T> Image< double > SurpriseMap< T >::surprise (const SurpriseImage< T > & sample)`

Update by receiving a new sample and sample measurement error.

The returned image is the surprise (combined temporal and spatial for each map, then combined over all maps) resulting from that update

Definition at line 81 of file SurpriseMap.C.

References Image< T >::clear(), Image< T >::getDims(), Image< T >::getHeight(), SurpriseImage< T >::getMean(), Image< float >::getMinMax(), Image< T >::getVal(), Image< double >::getVal(), SurpriseImage< T >::getVar(), Image< T >::getWidth(), SurpriseImage< T >::neighborhoods(), NO\_INIT, Image< T >::rectify(), Image< float >::rectify(), SurpriseImage< T >::reset(), SurpriseImage< T >::resetUpdFac(), Image< T >::resize(), SurpriseImage< T >::surprise(), toPower(), and uint.

The documentation for this class was generated from the following files:

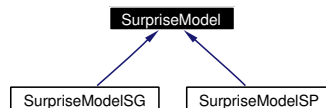
- [SurpriseMap.H](#)
- [SurpriseMap.C](#)

## 10.288 SurpriseModel Class Reference

A local (single-point) surprise model.

```
#include <SurpriseModel.H>
```

Inheritance diagram for SurpriseModel:



### Public Member Functions

- [SurpriseModel](#) (const double updatefac=0.5, const double sampleval=0.0, const double samplevar=1.0)  
*Constructor.*
- virtual [~SurpriseModel](#) ()  
*Virtual destructor ensures proper destruction of derived classes.*
- virtual void [reset](#) ()=0  
*Reset to initial state.*
- virtual void [init](#) (const double updatefac=0.5, const double sampleval=0.0, const double samplevar=1.0)  
*Change initial state parameters.*
- virtual void [resetUpdFac](#) (const double updfac)  
*Reset our update factor.*
- virtual void [load](#) (const double sampleval=0.0, const double samplevar=1.0)=0  
*Load given sample mean and variance.*
- virtual double [surprise](#) (const [SurpriseModel](#) &sample)  
*Compute surprise between us and another model.*
- virtual void [combineFrom](#) (const [SurpriseImage](#)< [SurpriseModel](#) > &models, const [Image](#)< float > &weights)  
*Initialize us as a weighted combination of the given map of models.*

- virtual void `combineFrom` (const `SurpriseImage`< `SurpriseModel` > &models, const `Image`< float > &weights, const `Point2D` &pos, const float wmin=0.0f)  
*Initialize us as a weighted combination of the given map of models.*
- virtual double `getMean` () const=0  
*get our mean*
- virtual double `getVar` () const=0  
*get our variance*

## Protected Attributes

- double `itsUpdateFac`  
*our update factor*
- double `itsInitialVal`  
*our initial value*
- double `itsInitialVar`  
*our initial variance*

### 10.288.1 Detailed Description

A local (single-point) surprise model.

This the base class and it cannot be implemented as some of its functions are purely virtual. It defines the basic interface. See the derivative classes for actual implementations.

Definition at line 49 of file `SurpriseModel.H`.

### 10.288.2 Constructor & Destructor Documentation

#### 10.288.2.1 `SurpriseModel::SurpriseModel` (const double *updatefac* = 0.5, const double *sampleval* = 0.0, const double *samplevar* = 1.0)

Constructor.

Build a new `SurpriseModel` object and initialize it as if an infinitely long sequence of samples had been previously received by the model.



**Parameters:**

- updatefac** factor by which the current model is combine with a new model on an update. Valid values are in [0..1]
- sampleval** the value of the samples previously received.
- samplevar** a variance expressing the intrinsic variability in the measurement of those samples.

Definition at line 44 of file SurpriseModel.C.

**10.288.2.2 SurpriseModel::~~SurpriseModel () [virtual]**

Virtual destructor ensures proper destruction of derived classes.

Definition at line 50 of file SurpriseModel.C.

**10.288.3 Member Function Documentation****10.288.3.1 void SurpriseModel::combineFrom (const SurpriseImage< SurpriseModel > & models, const Image< float > & weights, const Point2D & pos, const float wmin = 0.0f) [virtual]**

Initialize us as a weighted combination of the given map of models.

**Parameters:**

- models** an array of models, one model at each pixel of the image.
- weights** the weight to be given to each model. Weights should sum to 1.0 and this is not checked for here for computational efficiency. In this version, the weight image should have double+1 the width and height of the Surprise image, and the weight for the current point will be taken at the center of the weight map, etc.
- pos** indicates the 2D coordinates of the center of the neighborhood in the Surprise-Image
- minw** is the minimum weight below which we just don't include a model in the update

Definition at line 73 of file SurpriseModel.C.

**10.288.3.2 void SurpriseModel::combineFrom (const SurpriseImage< SurpriseModel > & models, const Image< float > & weights) [virtual]**

Initialize us as a weighted combination of the given map of models.

**Parameters:**

***models*** an array of models, one model at each pixel of the image.

***weights*** the weight to be given to each model. Weights should sum to 1.0 and this is not checked for here for computational efficiency.

Definition at line 68 of file SurpriseModel.C.

**10.288.3.3** **virtual double SurpriseModel::getMean () const** [pure virtual]

get our mean

Implemented in [SurpriseModelSG](#), and [SurpriseModelSP](#).

**10.288.3.4** **virtual double SurpriseModel::getVar () const** [pure virtual]

get our variance

Implemented in [SurpriseModelSG](#), and [SurpriseModelSP](#).

**10.288.3.5** **void SurpriseModel::init (const double *updatefac* = 0.5, const double *sampleval* = 0.0, const double *samplevar* = 1.0)** [virtual]

Change initial state parameters.

Reimplemented in [SurpriseModelSG](#), and [SurpriseModelSP](#).

Definition at line 54 of file SurpriseModel.C.

Referenced by [SurpriseModelSP::init\(\)](#), and [SurpriseModelSG::init\(\)](#).

**10.288.3.6** **virtual void SurpriseModel::load (const double *sampleval* = 0.0, const double *samplevar* = 1.0)** [pure virtual]

Load given sample mean and variance.

Implemented in [SurpriseModelSG](#), and [SurpriseModelSP](#).

**10.288.3.7** **virtual void SurpriseModel::reset ()** [pure virtual]

Reset to initial state.

Implemented in [SurpriseModelSG](#), and [SurpriseModelSP](#).

**10.288.3.8 void SurpriseModel::resetUpdFac (const double *updfac*)**  
[virtual]

Reset our update factor.

Definition at line 60 of file SurpriseModel.C.

**10.288.3.9 double SurpriseModel::surprise (const [SurpriseModel](#) & *sample*)**  
[virtual]

Compute surprise between us and another model.

Surprise is the KL distance between our current model and what that model will become once we update it with the sample passed in as argument. For the update, our current model is weighted by itsUpdateFac while the new model by (1.0-itsUpdateFac). Programmer note: derived classes will not provide a direct overload of this (and other) base functions, as instead they will implement functions with basically the same syntax but using an argument of the derived class. Since an implementation is not provided for this base function, you will get an error at link time if you call the derived functions with a base-class argument.

Definition at line 64 of file SurpriseModel.C.

**10.288.4 Member Data Documentation****10.288.4.1 double [SurpriseModel::itsInitialVal](#)** [protected]

our initial value

Definition at line 127 of file SurpriseModel.H.

**10.288.4.2 double [SurpriseModel::itsInitialVar](#)** [protected]

our initial variance

Definition at line 128 of file SurpriseModel.H.

**10.288.4.3 double [SurpriseModel::itsUpdateFac](#)** [protected]

our update factor

Definition at line 126 of file SurpriseModel.H.

The documentation for this class was generated from the following files:

- [SurpriseModel.H](#)

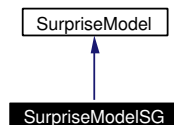
- [SurpriseModel.C](#)

## 10.289 SurpriseModelSG Class Reference

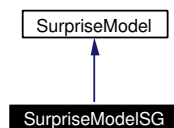
A single-Gaussian [SurpriseModel](#).

```
#include <SurpriseModel.H>
```

Inheritance diagram for SurpriseModelSG:



Collaboration diagram for SurpriseModelSG:



### Public Member Functions

- [SurpriseModelSG](#) (const double updatefac=0.5, const double sampleval=0.0, const double samplevar=1.0)  
*Constructor. See base class for details.*
- virtual [~SurpriseModelSG](#) ()  
*Virtual destructor ensures proper destruction of derived classes.*
- virtual void [reset](#) ()  
*Reset to initial state.*
- virtual void [init](#) (const double updatefac=0.5, const double sampleval=0.0, const double samplevar=1.0)  
*Change initial state parameters.*
- virtual void [load](#) (const double sampleval, const double samplevar=1.0)  
*Load given sample mean and variance.*
- virtual double [surprise](#) (const [SurpriseModelSG](#) &other)

*Compute surprise between us and another model.*

- virtual void `combineFrom` (const `SurpriseImage`< `SurpriseModelSG` > &models, const `Image`< float > &weights)

*Initialize us as a weighted combination of the given map of models.*

- virtual void `combineFrom` (const `SurpriseImage`< `SurpriseModelSG` > &models, const `Image`< float > &weights, const `Point2D` &pos, const float wmin=0.0f)

*Initialize us as a weighted combination of the given map of models.*

- virtual double `getMean` () const

*get our mean*

- virtual double `getVar` () const

*get our variance*

## Protected Attributes

- double `itsMean`

*our current Gaussian mean*

- double `itsVar`

*our current Gaussian variance*

### 10.289.1 Detailed Description

A single-Gaussian `SurpriseModel`.

This is a very simple `SurpriseModel` consisting of a single Gaussian, which is updated in a sliding-average manner.

Definition at line 135 of file `SurpriseModel.H`.

### 10.289.2 Constructor & Destructor Documentation

#### 10.289.2.1 `SurpriseModelSG::SurpriseModelSG` (const double *updatefac* = 0.5, const double *sampleval* = 0.0, const double *samplevar* = 1.0)

Constructor. See base class for details.

Definition at line 84 of file `SurpriseModel.C`.

### 10.289.2.2 SurpriseModelSG::~~SurpriseModelSG () [virtual]

Virtual destructor ensures proper destruction of derived classes.

Definition at line 93 of file SurpriseModel.C.

## 10.289.3 Member Function Documentation

### 10.289.3.1 void SurpriseModelSG::combineFrom (const SurpriseImage< SurpriseModelSG > & models, const Image< float > & weights, const Point2D & pos, const float wmin = 0.0f) [virtual]

Initialize us as a weighted combination of the given map of models.

Definition at line 169 of file SurpriseModel.C.

References ASSERT, Image< T >::begin(), Image< T >::coordsOk(), Image< T >::getHeight(), Image< T >::getWidth(), Point2D::i, itsVar, and Point2D::j.

### 10.289.3.2 void SurpriseModelSG::combineFrom (const SurpriseImage< SurpriseModelSG > & models, const Image< float > & weights) [virtual]

Initialize us as a weighted combination of the given map of models.

Definition at line 143 of file SurpriseModel.C.

References ASSERT, Image< T >::begin(), Image< T >::end(), Image< T >::isSameSize(), and itsVar.

### 10.289.3.3 double SurpriseModelSG::getMean () const [virtual]

get our mean

Implements [SurpriseModel](#).

Definition at line 209 of file SurpriseModel.C.

### 10.289.3.4 double SurpriseModelSG::getVar () const [virtual]

get our variance

Implements [SurpriseModel](#).

Definition at line 213 of file SurpriseModel.C.

References itsVar.

**10.289.3.5** `void SurpriseModelSG::init (const double updatefac = 0.5, const double sampleval = 0.0, const double samplevar = 1.0) [virtual]`

Change initial state parameters.

Reimplemented from [SurpriseModel](#).

Definition at line 101 of file SurpriseModel.C.

References [SurpriseModel::init\(\)](#), and [reset\(\)](#).

**10.289.3.6** `void SurpriseModelSG::load (const double sampleval, const double samplevar = 1.0) [virtual]`

Load given sample mean and variance.

Implements [SurpriseModel](#).

Definition at line 110 of file SurpriseModel.C.

References [itsVar](#).

Referenced by [reset\(\)](#).

**10.289.3.7** `void SurpriseModelSG::reset () [virtual]`

Reset to initial state.

Implements [SurpriseModel](#).

Definition at line 97 of file SurpriseModel.C.

References [load\(\)](#).

Referenced by [init\(\)](#).

**10.289.3.8** `double SurpriseModelSG::surprise (const SurpriseModelSG & other) [virtual]`

Compute surprise between us and another model.

Definition at line 115 of file SurpriseModel.C.

References [itsMean](#), and [itsVar](#).



## 10.289.4 Member Data Documentation

### 10.289.4.1 double [SurpriseModelSG::itsMean](#) [protected]

our current Gaussian mean

Definition at line 174 of file SurpriseModel.H.

Referenced by surprise().

### 10.289.4.2 double [SurpriseModelSG::itsVar](#) [protected]

our current Gaussian variance

Definition at line 175 of file SurpriseModel.H.

Referenced by combineFrom(), getVar(), load(), and surprise().

The documentation for this class was generated from the following files:

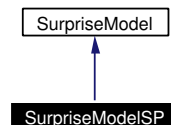
- [SurpriseModel.H](#)
- [SurpriseModel.C](#)

## 10.290 SurpriseModelSP Class Reference

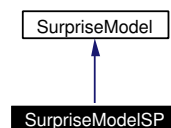
A single-Poisson/Gamma [SurpriseModel](#).

```
#include <SurpriseModel.H>
```

Inheritance diagram for SurpriseModelSP:



Collaboration diagram for SurpriseModelSP:



### Public Member Functions

- [SurpriseModelSP](#) (const double updatefac=0.5, const double sampleval=0.0, const double samplevar=1.0)  
*Constructor. See base class for details.*
- virtual [~SurpriseModelSP](#) ()  
*Virtual destructor ensures proper destruction of derived classes.*
- virtual void [reset](#) ()  
*Reset to initial state.*
- virtual void [init](#) (const double updatefac=0.5, const double sampleval=0.0, const double samplevar=1.0)  
*Change initial state parameters.*
- virtual void [load](#) (const double sampleval, const double samplevar=1.0)  
*Load given sample mean and variance.*
- virtual double [surprise](#) (const [SurpriseModelSP](#) &other)

*Compute surprise between us and another model.*

- virtual void [combineFrom](#) (const [SurpriseImage](#)< [SurpriseModelSP](#) > &models, const [Image](#)< float > &weights)

*Initialize us as a weighted combination of the given map of models.*

- virtual void [combineFrom](#) (const [SurpriseImage](#)< [SurpriseModelSP](#) > &models, const [Image](#)< float > &weights, const [Point2D](#) &pos, const float wmin=0.0f)

*Initialize us as a weighted combination of the given map of models.*

- virtual double [getMean](#) () const

*get our mean*

- virtual double [getVar](#) () const

*get our variance*

## Protected Attributes

- int [itsN](#)

*our sample N*

- double [itsAlpha](#)

*our current Gamma alpha*

- double [itsBeta](#)

*our current Gamma beta*

### 10.290.1 Detailed Description

A single-Poisson/Gamma [SurpriseModel](#).

This is a very simple [SurpriseModel](#) consisting of a single Gamma prior over Poisson, which is updated in a sliding-average manner.

Definition at line 182 of file SurpriseModel.H.

## 10.290.2 Constructor & Destructor Documentation

### 10.290.2.1 `SurpriseModelSP::SurpriseModelSP (const double updatefac = 0.5, const double sampleval = 0.0, const double samplevar = 1.0)`

Constructor. See base class for details.

Definition at line 220 of file `SurpriseModel.C`.

### 10.290.2.2 `SurpriseModelSP::~~SurpriseModelSP ()` `[virtual]`

Virtual destructor ensures proper destruction of derived classes.

Definition at line 230 of file `SurpriseModel.C`.

## 10.290.3 Member Function Documentation

### 10.290.3.1 `void SurpriseModelSP::combineFrom (const SurpriseImage< SurpriseModelSP > & models, const Image< float > & weights, const Point2D & pos, const float wmin = 0.0f)` `[virtual]`

Initialize us as a weighted combination of the given map of models.

Definition at line 308 of file `SurpriseModel.C`.

References `ASSERT`, `Image< T >::begin()`, `Image< T >::coordsOk()`, `Image< T >::getHeight()`, `Image< T >::getWidth()`, `Point2D::i`, `itsAlpha`, `itsBeta`, and `Point2D::j`.

### 10.290.3.2 `void SurpriseModelSP::combineFrom (const SurpriseImage< SurpriseModelSP > & models, const Image< float > & weights)` `[virtual]`

Initialize us as a weighted combination of the given map of models.

Definition at line 281 of file `SurpriseModel.C`.

References `ASSERT`, `Image< T >::begin()`, `Image< T >::end()`, `Image< T >::isSameSize()`, `itsAlpha`, and `itsBeta`.

### 10.290.3.3 `double SurpriseModelSP::getMean ()` `const` `[virtual]`

get our mean

Implements [SurpriseModel](#).

Definition at line 349 of file `SurpriseModel.C`.

References itsAlpha, and itsBeta.

#### 10.290.3.4 double SurpriseModelSP::getVar () const [virtual]

get our variance

Implements [SurpriseModel](#).

Definition at line 353 of file SurpriseModel.C.

References itsAlpha, and itsBeta.

#### 10.290.3.5 void SurpriseModelSP::init (const double *updatefac* = 0.5, const double *sampleval* = 0.0, const double *samplevar* = 1.0) [virtual]

Change intial state parameters.

Reimplemented from [SurpriseModel](#).

Definition at line 238 of file SurpriseModel.C.

References SurpriseModel::init(), and load().

#### 10.290.3.6 void SurpriseModelSP::load (const double *sampleval*, const double *samplevar* = 1.0) [virtual]

Load given sample mean and variance.

Implements [SurpriseModel](#).

Definition at line 247 of file SurpriseModel.C.

References itsAlpha, itsBeta, and itsN.

Referenced by init(), and reset().

#### 10.290.3.7 void SurpriseModelSP::reset () [virtual]

Reset to initial state.

Implements [SurpriseModel](#).

Definition at line 234 of file SurpriseModel.C.

References load().

**10.290.3.8 double SurpriseModelSP::surprise (const [SurpriseModelSP](#) & other) [virtual]**

Compute surprise between us and another model.

Definition at line 255 of file SurpriseModel.C.

References itsAlpha, itsBeta, itsN, and psi().

**10.290.4 Member Data Documentation****10.290.4.1 double [SurpriseModelSP::itsAlpha](#) [protected]**

our current Gamma alpha

Definition at line 222 of file SurpriseModel.H.

Referenced by combineFrom(), getMean(), getVar(), load(), and surprise().

**10.290.4.2 double [SurpriseModelSP::itsBeta](#) [protected]**

our current Gamma beta

Definition at line 223 of file SurpriseModel.H.

Referenced by combineFrom(), getMean(), getVar(), load(), and surprise().

**10.290.4.3 int [SurpriseModelSP::itsN](#) [protected]**

our sample N

Definition at line 221 of file SurpriseModel.H.

Referenced by load(), and surprise().

The documentation for this class was generated from the following files:

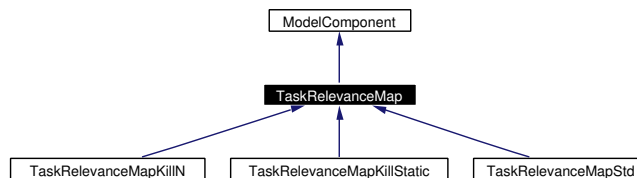
- [SurpriseModel.H](#)
- [SurpriseModel.C](#)

## 10.291 TaskRelevanceMap Class Reference

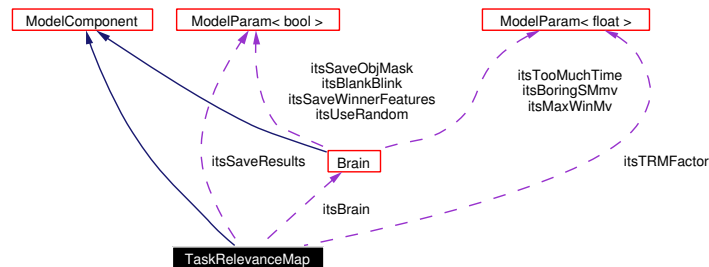
The task-relevance map base class.

```
#include <TaskRelevanceMap.H>
```

Inheritance diagram for TaskRelevanceMap:



Collaboration diagram for TaskRelevanceMap:



## Public Member Functions

### Constructor, destructor, and reset

- `TaskRelevanceMap` (`ModelManager` &mgr, const `std::string` &descrName="Task-Relevance Map", const `std::string` &tagName="TaskRelevance-Map")  
*Initialized constructor.*
- virtual void `exportOptions` (const int whatToExport, const bool recurse=true)  
*Export some of our `ModelParam` parameters as command-line-options.*
- virtual `~TaskRelevanceMap` ()  
*Destructor.*

- virtual void [reset](#) ()=0  
*Reset to initial state just after construction.*
- void [setBrain](#) ([Brain](#) \*brain)  
*Setup a hook back to our brain.*

### Input, evolve and output

- virtual void [input](#) (const [Image](#)< float > &in)=0  
*Set new relevance values.*
- virtual int [getWidth](#) () const=0  
*Get width of the map.*
- virtual int [getHeight](#) () const=0  
*Get height of the map.*
- virtual [Image](#)< float > [getV](#) (const bool normalize=false) const=0  
*Return all our relevance values as an [Image](#)<float>.*
- virtual float [getV](#) (const [Point2D](#) &p) const=0  
*Access function using [Point2D](#).*
- virtual void [evolve](#) (const double t)=0  
*Integrate inputs until time t (in s) and update internal state.*
- virtual void [update](#) (const [Point2D](#) &fixation, const float relevance)=0  
*update TRM with relevance of current fixation*

### Results

- virtual void [saveResults](#) (const [SharedPtr](#)< [OutputFrameSeries](#) > &ofs)  
*Save our various results.*

### Protected Attributes

- [ModelParam](#)< bool > [itsSaveResults](#)  
*Save our internals when [saveResults\(\)](#) is called?*
- [ModelParam](#)< float > [itsTRMFactor](#)  
*Multiply by a factor rather than normalize when doing [getV\(\)](#)?*
- [Brain](#) \* [itsBrain](#)  
*Our brain, or [SharedPtr](#)<NULL> if it has not been set with [setBrain\(\)](#).*



### 10.291.1 Detailed Description

The task-relevance map base class.

This is a 2D task-relevance map. It is just a base class with virtual function definitions. Various implementations are available below. The TRM is a topographic map where a value of zero indicates that a location is not relevant to the task at hand, while a value of 1.0 indicates neutral relevance and values higher than 1.0 indicate high relevance.

Definition at line 58 of file TaskRelevanceMap.H.

### 10.291.2 Constructor & Destructor Documentation

**10.291.2.1** `TaskRelevanceMap::TaskRelevanceMap (ModelManager & mgr, const std::string & descrName = "Task-Relevance Map", const std::string & tagName = "TaskRelevanceMap")`

Initialized constructor.

The map will be resized and initialized the first time [input\(\)](#) is called

Definition at line 59 of file TaskRelevanceMap.C.

**10.291.2.2** `TaskRelevanceMap::~TaskRelevanceMap ()` [virtual]

Destructor.

Definition at line 69 of file TaskRelevanceMap.C.

### 10.291.3 Member Function Documentation

**10.291.3.1** `virtual void TaskRelevanceMap::evolve (const double t)` [pure virtual]

Integrate inputs until time t (in s) and update internal state.

Implemented in [TaskRelevanceMapStd](#), [TaskRelevanceMapKillStatic](#), and [TaskRelevanceMapKillN](#).

**10.291.3.2** `void TaskRelevanceMap::exportOptions (const int whatToExport, const bool recurse = true)` [virtual]

Export some of our [ModelParam](#) parameters as command-line-options.

See the base function in [ModelComponent.H](#) for info. A base implementation is provided that will only export the base class options (see the [ModelParam](#) definitions in

the protected section below).

Reimplemented from [ModelComponent](#).

Reimplemented in [TaskRelevanceMapStd](#), [TaskRelevanceMapKillStatic](#), and [TaskRelevanceMapKillN](#).

Definition at line 73 of file TaskRelevanceMap.C.

References [ModelComponent::exportOptions\(\)](#), [itsTRMFactor](#), [OPTEXP\\_CORE](#), [OPTEXP\\_SAVE](#), and [ModelManager::requestOption\(\)](#).

Referenced by [TaskRelevanceMapKillN::exportOptions\(\)](#), [TaskRelevanceMapKillStatic::exportOptions\(\)](#), and [TaskRelevanceMapStd::exportOptions\(\)](#).

#### 10.291.3.3 **virtual int TaskRelevanceMap::getHeight () const** [pure virtual]

Get height of the map.

Implemented in [TaskRelevanceMapStd](#), [TaskRelevanceMapKillStatic](#), and [TaskRelevanceMapKillN](#).

#### 10.291.3.4 **virtual float TaskRelevanceMap::getV (const [Point2D](#) & p) const** [pure virtual]

Access function using [Point2D](#).

Implemented in [TaskRelevanceMapStd](#), [TaskRelevanceMapKillStatic](#), and [TaskRelevanceMapKillN](#).

#### 10.291.3.5 **virtual [Image](#)<float> TaskRelevanceMap::getV (const bool *normalize* = false) const** [pure virtual]

Return all our relevance values as an [Image](#)<float>.

The current values are possibly normalized to 0..255. The normalization will use its-TRMFactor if it is non-null

Implemented in [TaskRelevanceMapStd](#), [TaskRelevanceMapKillStatic](#), and [TaskRelevanceMapKillN](#).

Referenced by [saveResults\(\)](#).

#### 10.291.3.6 **virtual int TaskRelevanceMap::getWidth () const** [pure virtual]

Get width of the map.

Implemented in [TaskRelevanceMapStd](#), [TaskRelevanceMapKillStatic](#), and [TaskRelevanceMapKillN](#).

#### 10.291.3.7 **virtual void TaskRelevanceMap::input (const [Image](#)< float > & in)** [pure virtual]

Set new relevance values.

This should initialize and resize the map if it is currently uninitialized (e.g., just after construction or [reset\(\)](#)).

Implemented in [TaskRelevanceMapStd](#), [TaskRelevanceMapKillStatic](#), and [TaskRelevanceMapKillN](#).

#### 10.291.3.8 **virtual void TaskRelevanceMap::reset ()** [pure virtual]

Reset to initial state just after construction.

Implemented in [TaskRelevanceMapStd](#), [TaskRelevanceMapKillStatic](#), and [TaskRelevanceMapKillN](#).

#### 10.291.3.9 **void TaskRelevanceMap::saveResults (const [SharedPtr](#)< [OutputFrameSeries](#) > & ofs)** [virtual]

Save our various results.

A default implementation is provided, which just calls [getV\(\)](#) and saves that map with a file name prefix of "SM".

#### Parameters:

*ofs* will figure out file name/format and save our results.

Definition at line 88 of file TaskRelevanceMap.C.

References [getV\(\)](#), and [ModelParam< bool >::getVal\(\)](#).

#### 10.291.3.10 **void TaskRelevanceMap::setBrain ([Brain](#) \* brain)**

Setup a hook back to our brain.

Definition at line 95 of file TaskRelevanceMap.C.

#### 10.291.3.11 **virtual void TaskRelevanceMap::update (const [Point2D](#) & fixation, const float *relevance*)** [pure virtual]

update TRM with relevance of current fixation

**Parameters:**

*fixation* enter the current fixation point

*relevance* enter the relevance of current fixation

Implemented in [TaskRelevanceMapStd](#), [TaskRelevanceMapKillStatic](#), and [TaskRelevanceMapKillN](#).

## 10.291.4 Member Data Documentation

### 10.291.4.1 [Brain\\*](#) [TaskRelevanceMap::itsBrain](#) [protected]

Our brain, or `SharedPtr<NULL>` if it has not been set with [setBrain\(\)](#).

Definition at line 146 of file `TaskRelevanceMap.H`.

### 10.291.4.2 [ModelParam<bool>](#) [TaskRelevanceMap::itsSaveResults](#) [protected]

Save our internals when [saveResults\(\)](#) is called?

Definition at line 140 of file `TaskRelevanceMap.H`.

### 10.291.4.3 [ModelParam<float>](#) [TaskRelevanceMap::itsTRMFactor](#) [protected]

Multiply by a factor rather than normalize when doing [getV\(\)](#)?

Definition at line 143 of file `TaskRelevanceMap.H`.

Referenced by [exportOptions\(\)](#).

The documentation for this class was generated from the following files:

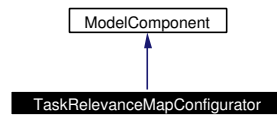
- [TaskRelevanceMap.H](#)
- [TaskRelevanceMap.C](#)

## 10.292 TaskRelevanceMapConfigurator Class Reference

[TaskRelevanceMap](#) configurator.

```
#include <TaskRelevanceMap.H>
```

Inheritance diagram for TaskRelevanceMapConfigurator:



Collaboration diagram for TaskRelevanceMapConfigurator:

### Public Member Functions

- [TaskRelevanceMapConfigurator](#) ([ModelManager](#) &mgr, const [std::string](#) &descrName="Task-Relevance Map Configurator", const [std::string](#) &tagName="TaskRelevanceMapConfigurator")

*Constructor.*

- virtual [~TaskRelevanceMapConfigurator](#) ()

*destructor*

- virtual void [exportOptions](#) (const int whatToExport, const bool recurse=true)

*Export some of our [ModelParam](#) parameters as command-line-options.*

- [SharedPtr](#)< [TaskRelevanceMap](#) > [getTRM](#) () const

*Get the chosen TRM.*

### Protected Member Functions

- virtual bool [setModelParamString](#) (const [std::string](#) &name, const [std::string](#) &value, const bool recurse=false, const bool warn=true)

*Intercept people changing our [ModelParam](#).*

## Protected Attributes

- [ModelParam](#) < [std::string](#) > [itsTRMtype](#)  
*type of map*

### 10.292.1 Detailed Description

[TaskRelevanceMap](#) configurator.

This will export the `-trm-type=XX` command-line option and will instantiate a TRM of the desired type as the option gets assigned a value. As this happens, new options may become available in the command-line. To see them, use `-help` AFTER you have chosen the type to use. The current TRM may be retrieved using [getTRM\(\)](#).

Definition at line 162 of file [TaskRelevanceMap.H](#).

### 10.292.2 Constructor & Destructor Documentation

**10.292.2.1** [TaskRelevanceMapConfigurator::TaskRelevanceMapConfigurator](#) ([ModelManager](#) & *mgr*, const [std::string](#) & *descrName* = "Task-Relevance Map Configurator", const [std::string](#) & *tagName* = "TaskRelevanceMapConfigurator")

Constructor.

Definition at line 104 of file [TaskRelevanceMap.C](#).

References [OPTEXP\\_NONE](#).

**10.292.2.2** [TaskRelevanceMapConfigurator::~~TaskRelevanceMapConfigurator](#) () [virtual]

destructor

Definition at line 113 of file [TaskRelevanceMap.C](#).

### 10.292.3 Member Function Documentation

**10.292.3.1** [void TaskRelevanceMapConfigurator::exportOptions](#) (const int *whatToExport*, const bool *recurse* = true) [virtual]

Export some of our [ModelParam](#) parameters as command-line-options.

See the base function in [ModelComponent.H](#) for info. Actually, here we just keep a copy of whatToExport and we will call [exportOptions\(\)](#) on our SM once we know its type and instantiate it.

Reimplemented from [ModelComponent](#).

Definition at line 117 of file TaskRelevanceMap.C.

References [ModelComponent::exportOptions\(\)](#), [itsTRMtype](#), [OPTEXP\\_CORE](#), and [ModelManager::requestOption\(\)](#).

### 10.292.3.2 [SharedPtr< TaskRelevanceMap > TaskRelevanceMapConfigurator::getTRM \(\) const](#)

Get the chosen TRM.

You should call this during [start\(\)](#) of the [ModelComponent](#) that needs the TRM. CAUTION: this may be a [SharedPtr\(NULL\)](#) if no TRM type has been chosen.

Definition at line 133 of file TaskRelevanceMap.C.

### 10.292.3.3 [bool TaskRelevanceMapConfigurator::setModelParamString \(const std::string & name, const std::string & value, const bool recurse = false, const bool warn = true\) \[protected, virtual\]](#)

Intercept people changing our [ModelParam](#).

See [ModelComponent.H](#); as parsing the command-line or reading a config file sets our name, we'll also here instantiate a controller of the proper type (and export its options)

Reimplemented from [ModelComponent](#).

Definition at line 138 of file TaskRelevanceMap.C.

References [ModelComponent::addSubComponent\(\)](#), [SharedPtr< TaskRelevanceMap >::get\(\)](#), [itsTRMtype](#), [ModelComponent::removeSubComponent\(\)](#), [SharedPtr< TaskRelevanceMap >::reset\(\)](#), and [ModelComponent::setModelParamString\(\)](#).

## 10.292.4 Member Data Documentation

### 10.292.4.1 [ModelParam<std::string> TaskRelevanceMapConfigurator::itsTRMtype \[protected\]](#)

type of map

Definition at line 190 of file TaskRelevanceMap.H.

Referenced by [exportOptions\(\)](#), and [setModelParamString\(\)](#).

The documentation for this class was generated from the following files:

- [TaskRelevanceMap.H](#)
- [TaskRelevanceMap.C](#)

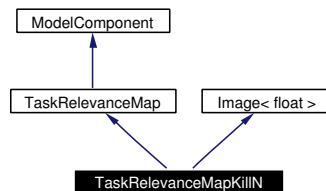


## 10.293 TaskRelevanceMapKillN Class Reference

A KillN task-relevance map.

```
#include <TaskRelevanceMap.H>
```

Inheritance diagram for TaskRelevanceMapKillN:



Collaboration diagram for TaskRelevanceMapKillN:

### Public Member Functions

- **TaskRelevanceMapKillN** (**ModelManager** &mgr, const **std::string** &descr-Name="Task-Relevance Map KillN", const **std::string** &tagName="Task-RelevanceMapKillN")  
*Uninitialized constructor.*
- virtual void **exportOptions** (const int whatToExport, const bool recurse=true)  
*Export some of our **ModelParam** parameters as command-line-options.*
- virtual **~TaskRelevanceMapKillN** ()  
*Destructor.*
- virtual void **reset** ()  
*Reset to initial state just after construction.*
- virtual void **input** (const **Image**< float > &in)  
*Set new relevance values.*
- virtual int **getWidth** () const  
*Get width of the map.*
- virtual int **getHeight** () const  
*Get height of the map.*

- virtual [Image](#)< float > [getV](#) (const bool normalize=false) const  
*Return all our relevance values as an [Image](#)<float>.*
- virtual float [getV](#) (const [Point2D](#) &p) const  
*Access function using [Point2D](#).*
- virtual void [evolve](#) (const double t)  
*Integrate inputs until time t (in s) and update internal state.*
- virtual void [update](#) (const [Point2D](#) &fixation, const float relevance)  
*update TRM with relevance of current fixation*

## Protected Attributes

- [ModelParam](#)< int > [itsN](#)  
*Number of inputs to compute over.*

### 10.293.1 Detailed Description

A KillN task-relevance map.

This is an alternative to IOR, in which we set the relevance of the last N attended locations to zero.

Definition at line 343 of file TaskRelevanceMap.H.

### 10.293.2 Constructor & Destructor Documentation

#### 10.293.2.1 TaskRelevanceMapKillN::TaskRelevanceMapKillN ([ModelManager](#) & mgr, const [std::string](#) & descrName = "Task-Relevance Map KillN", const [std::string](#) & tagName = "TaskRelevanceMapKillN")

Uninitialized constructor.

Definition at line 455 of file TaskRelevanceMap.C.

#### 10.293.2.2 TaskRelevanceMapKillN::~TaskRelevanceMapKillN () [virtual]

Destructor.

Definition at line 475 of file TaskRelevanceMap.C.

### 10.293.3 Member Function Documentation

#### 10.293.3.1 void TaskRelevanceMapKillN::evolve (const double t) [virtual]

Integrate inputs until time t (in s) and update internal state.

Implements [TaskRelevanceMap](#).

Definition at line 521 of file TaskRelevanceMap.C.

#### 10.293.3.2 void TaskRelevanceMapKillN::exportOptions (const int whatToExport, const bool recurse = true) [virtual]

Export some of our [ModelParam](#) parameters as command-line-options.

Reimplemented from [TaskRelevanceMap](#).

Definition at line 462 of file TaskRelevanceMap.C.

References [TaskRelevanceMap::exportOptions\(\)](#), [OPTEXP\\_CORE](#), and [ModelManager::requestOption\(\)](#).

#### 10.293.3.3 int TaskRelevanceMapKillN::getHeight () const [virtual]

Get height of the map.

Implements [TaskRelevanceMap](#).

Definition at line 533 of file TaskRelevanceMap.C.

References [Image< T >::getHeight\(\)](#).

#### 10.293.3.4 float TaskRelevanceMapKillN::getV (const [Point2D](#) & p) const [virtual]

Access function using [Point2D](#).

Implements [TaskRelevanceMap](#).

Definition at line 525 of file TaskRelevanceMap.C.

References [Image< float >::getVal\(\)](#).

#### 10.293.3.5 [Image< float >](#) TaskRelevanceMapKillN::getV (const bool normalize = false) const [virtual]

Return all our relevance values as an [Image<float>](#).

Implements [TaskRelevanceMap](#).

Definition at line 509 of file TaskRelevanceMap.C.

References `Image< T >::clamp()`, and `ModelParam< float >::getVal()`.

#### 10.293.3.6 `int TaskRelevanceMapKillN::getWidth () const` [virtual]

Get width of the map.

Implements [TaskRelevanceMap](#).

Definition at line 529 of file TaskRelevanceMap.C.

References `Image< T >::getWidth()`.

#### 10.293.3.7 `void TaskRelevanceMapKillN::input (const Image< float > & in)` [virtual]

Set new relevance values.

The expected input here is an object mask, with zero values inside the attended object and one outside, at the scale and dims of the saliency map. Typically, this can be obtained from an object mask given by a [ShapeEstimator](#). After each input, our relevance map is updated as a compound over the last N objects (with zero relevance at those objects, and unit relevance outside, with fuzzy boundaries).

Implements [TaskRelevanceMap](#).

Definition at line 483 of file TaskRelevanceMap.C.

References `binaryReverse()`, `chamfer34()`, `Image< T >::clamp()`, `ImageCacheMinMax< float >::getMin()`, `Image< T >::getMinMax()`, `ModelParam< int >::getVal()`, `Image< float >::initialized()`, `ImageCache< float >::push_back()`, `ImageCache< float >::setMaxSize()`, and `Image< float >::swap()`.

#### 10.293.3.8 `void TaskRelevanceMapKillN::reset ()` [virtual]

Reset to initial state just after construction.

Implements [TaskRelevanceMap](#).

Definition at line 479 of file TaskRelevanceMap.C.

References `ImageCache< float >::clear()`, and `Image< float >::freeMem()`.

#### 10.293.3.9 `void TaskRelevanceMapKillN::update (const Point2D & fixation, const float relevance)` [virtual]

update TRM with relevance of current fixation

Not supported, will throw a fatal error if called.

Implements [TaskRelevanceMap](#).

Definition at line 537 of file TaskRelevanceMap.C.

## 10.293.4 Member Data Documentation

### 10.293.4.1 [ModelParam](#)<int> [TaskRelevanceMapKillN::itsN](#) [protected]

Number of inputs to compute over.

Definition at line 393 of file TaskRelevanceMap.H.

The documentation for this class was generated from the following files:

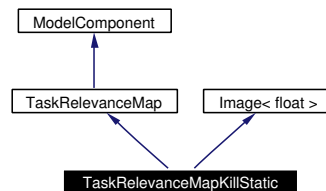
- [TaskRelevanceMap.H](#)
- [TaskRelevanceMap.C](#)

## 10.294 TaskRelevanceMapKillStatic Class Reference

A killStatic task-relevance map.

```
#include <TaskRelevanceMap.H>
```

Inheritance diagram for TaskRelevanceMapKillStatic:



Collaboration diagram for TaskRelevanceMapKillStatic:

### Public Member Functions

- [TaskRelevanceMapKillStatic](#) ([ModelManager](#) &mgr, const [std::string](#) &descr-Name="Task-Relevance Map KillStatic", const [std::string](#) &tagName="Task-RelevanceMapKillStatic")

*Uninitialized constructor.*

- virtual void [exportOptions](#) (const int whatToExport, const bool recurse=true)

*Export some of our [ModelParam](#) parameters as command-line-options.*

- virtual [~TaskRelevanceMapKillStatic](#) ()

*Destructor.*

- virtual void [reset](#) ()

*Reset to initial state just after construction.*

- virtual void [input](#) (const [Image](#)< float > &in)

*Set new relevance values.*

- virtual int [getWidth](#) () const

*Get width of the map.*

- virtual int [getHeight](#) () const

*Get height of the map.*

- virtual [Image](#)< float > [getV](#) (const bool normalize=false) const  
*Return all our relevance values as an [Image](#)<float>.*
- virtual float [getV](#) (const [Point2D](#) &p) const  
*Access function using [Point2D](#).*
- virtual void [evolve](#) (const double t)  
*Integrate inputs until time t (in s) and update internal state.*
- virtual void [update](#) (const [Point2D](#) &fixation, const float relevance)  
*update TRM with relevance of current fixation*

## Protected Member Functions

- virtual void [start1](#) ()  
*get ourselves started*

## Protected Attributes

- [ModelParam](#)< float > [itsKillStaticThresh](#)
- [ModelParam](#)< float > [itsKillStaticCoeff](#)
- [Image](#)< float > [itsStaticBuff](#)
- [SharedPtr](#)< [SaccadeController](#) > [itsSC](#)
- [SharedPtr](#)< [IntensityChannel](#) > [itsIntensChan](#)
- int [itsMapLevel](#)
- double [itsLastTime](#)

### 10.294.1 Detailed Description

A killStatic task-relevance map.

This is a very simple TRM where image locations that do not change much over time see their relevance progressively decrease.

Definition at line 270 of file TaskRelevanceMap.H.

## 10.294.2 Constructor & Destructor Documentation

**10.294.2.1** `TaskRelevanceMapKillStatic::TaskRelevanceMapKillStatic (ModelManager & mgr, const std::string & descrName = "Task-Relevance Map KillStatic", const std::string & tagName = "TaskRelevanceMapKillStatic")`

Uninitialized constructor.

Definition at line 315 of file TaskRelevanceMap.C.

**10.294.2.2** `TaskRelevanceMapKillStatic::~TaskRelevanceMapKillStatic () [virtual]`

Destructor.

Definition at line 339 of file TaskRelevanceMap.C.

## 10.294.3 Member Function Documentation

**10.294.3.1** `void TaskRelevanceMapKillStatic::evolve (const double t) [virtual]`

Integrate inputs until time t (in s) and update internal state.

Implements [TaskRelevanceMap](#).

Definition at line 380 of file TaskRelevanceMap.C.

References [absDiff\(\)](#), [Image< T >::clamp\(\)](#), [Image< float >::freeMem\(\)](#), [SharedPtr< SaccadeController >::get\(\)](#), [Image< T >::getMinMax\(\)](#), [ModelParam< float >::getVal\(\)](#), [Image< float >::initialized\(\)](#), [itsKillStaticCoeff](#), [itsKillStaticThresh](#), and [Image< float >::swap\(\)](#).

**10.294.3.2** `void TaskRelevanceMapKillStatic::exportOptions (const int whatToExport, const bool recurse = true) [virtual]`

Export some of our [ModelParam](#) parameters as command-line-options.

Reimplemented from [TaskRelevanceMap](#).

Definition at line 325 of file TaskRelevanceMap.C.

References [TaskRelevanceMap::exportOptions\(\)](#), [itsKillStaticCoeff](#), [itsKillStaticThresh](#), [OPTEXP\\_CORE](#), and [ModelManager::requestOption\(\)](#).



**10.294.3.3** `int TaskRelevanceMapKillStatic::getHeight () const` [virtual]

Get height of the map.

Implements [TaskRelevanceMap](#).

Definition at line 438 of file TaskRelevanceMap.C.

References `Image< T >::getHeight()`.

**10.294.3.4** `float TaskRelevanceMapKillStatic::getV (const Point2D & p) const` [virtual]

Access function using [Point2D](#).

Implements [TaskRelevanceMap](#).

Definition at line 430 of file TaskRelevanceMap.C.

References `Image< float >::getVal()`.

**10.294.3.5** `Image< float > TaskRelevanceMapKillStatic::getV (const bool normalize = false) const` [virtual]

Return all our relevance values as an [Image<float>](#).

Implements [TaskRelevanceMap](#).

Definition at line 368 of file TaskRelevanceMap.C.

References `Image< T >::clamp()`, and `ModelParam< float >::getVal()`.

**10.294.3.6** `int TaskRelevanceMapKillStatic::getWidth () const` [virtual]

Get width of the map.

Implements [TaskRelevanceMap](#).

Definition at line 434 of file TaskRelevanceMap.C.

References `Image< T >::getWidth()`.

**10.294.3.7** `void TaskRelevanceMapKillStatic::input (const Image< float > & in)` [virtual]

Set new relevance values.

Implements [TaskRelevanceMap](#).

Definition at line 364 of file TaskRelevanceMap.C.

References `Image< float >::swap()`.

#### 10.294.3.8 `void TaskRelevanceMapKillStatic::reset ()` [virtual]

Reset to initial state just after construction.

Implements [TaskRelevanceMap](#).

Definition at line 360 of file `TaskRelevanceMap.C`.

References `Image< float >::freeMem()`.

#### 10.294.3.9 `void TaskRelevanceMapKillStatic::start1 ()` [protected, virtual]

get ourselves started

Reimplemented from [ModelComponent](#).

Definition at line 343 of file `TaskRelevanceMap.C`.

References `SharedPtr< IntensityChannel >::dynCastFrom()`, `SharedPtr< T >::get()`, `SharedPtr< SaccadeController >::get()`, `Brain::getSC()`, `Brain::getVC()`, and `LevelSpec::mapLevel()`.

#### 10.294.3.10 `void TaskRelevanceMapKillStatic::update (const Point2D & fixation, const float relevance)` [virtual]

update TRM with relevance of current fixation

Not supported, will throw a fatal error if called.

Implements [TaskRelevanceMap](#).

Definition at line 442 of file `TaskRelevanceMap.C`.

### 10.294.4 Member Data Documentation

#### 10.294.4.1 `ModelParam<float> TaskRelevanceMapKillStatic::itsKillStaticCoeff` [protected]

Max killing achievable for a purely static location (should be in 0.0 .. 1.0 range)

Definition at line 322 of file `TaskRelevanceMap.H`.

Referenced by `evolve()`, and `exportOptions()`.

#### 10.294.4.2 [ModelParam<float> TaskRelevanceMapKillStatic::itsKillStaticThresh](#) [protected]

Anything in the difference between current and cumulated previous frames (range 0..255) less than this will be considered static

Definition at line 318 of file TaskRelevanceMap.H.

Referenced by [evolve\(\)](#), and [exportOptions\(\)](#).

The documentation for this class was generated from the following files:

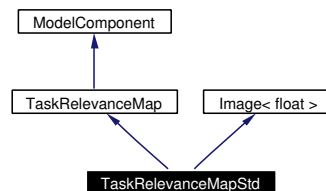
- [TaskRelevanceMap.H](#)
- [TaskRelevanceMap.C](#)

## 10.295 TaskRelevanceMapStd Class Reference

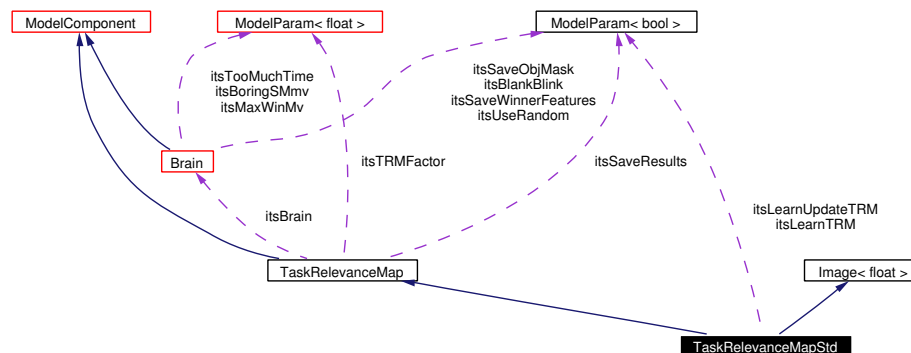
The standard task-relevance map.

```
#include <TaskRelevanceMap.H>
```

Inheritance diagram for TaskRelevanceMapStd:



Collaboration diagram for TaskRelevanceMapStd:



### Public Member Functions

- **TaskRelevanceMapStd** (**ModelManager** &mgr, const **std::string** &descr-Name="Task-Relevance Map Std", const **std::string** &tagName="Task-RelevanceMapStd")

*Uninitialized constructor.*

- virtual void **exportOptions** (const int whatToExport, const bool recurse=true)

*Export some of our **ModelParam** parameters as command-line-options.*

- virtual **~TaskRelevanceMapStd** ()

*Destructor.*

- virtual void [reset](#) ()  
*Reset to initial state just after construction.*
- virtual void [input](#) (const [Image](#)< float > &[in](#))  
*Set new relevance values.*
- virtual int [getWidth](#) () const  
*Get width of the map.*
- virtual int [getHeight](#) () const  
*Get height of the map.*
- virtual [Image](#)< float > [getV](#) (const bool normalize=false) const  
*Return all our relevance values as an [Image](#)<float>.*
- virtual float [getV](#) (const [Point2D](#) &p) const  
*Access function using [Point2D](#).*
- virtual void [evolve](#) (const double t)  
*Integrate inputs until time t (in s) and update internal state.*
- virtual void [update](#) (const [Point2D](#) &fixation, const float relevance)  
*update TRM with relevance of current fixation*

## Protected Attributes

- [ModelParam](#)< bool > [itsLearnTRM](#)
- [ModelParam](#)< bool > [itsLearnUpdateTRM](#)
- [ModelParam](#)< std::string > [itsBiasTRM](#)

### 10.295.1 Detailed Description

The standard task-relevance map.

This is our current standard TRM implementation. It can be loaded all at once using [input\(\)](#) or updated for a specific location using [update\(\)](#); [evolve\(\)](#) is a no-op. It's a pure spatial short-term memory.

Definition at line 214 of file TaskRelevanceMap.H.

## 10.295.2 Constructor & Destructor Documentation

### 10.295.2.1 TaskRelevanceMapStd::TaskRelevanceMapStd ([ModelManager](#) & *mgr*, const [std::string](#) & *descrName* = "Task-Relevance Map Std", const [std::string](#) & *tagName* = "TaskRelevanceMapStd")

Uninitialized constructor.

Definition at line 189 of file TaskRelevanceMap.C.

### 10.295.2.2 TaskRelevanceMapStd::~TaskRelevanceMapStd () [virtual]

Destructor.

Definition at line 213 of file TaskRelevanceMap.C.

## 10.295.3 Member Function Documentation

### 10.295.3.1 void TaskRelevanceMapStd::evolve (const double t) [virtual]

Integrate inputs until time t (in s) and update internal state.

Implements [TaskRelevanceMap](#).

Definition at line 250 of file TaskRelevanceMap.C.

### 10.295.3.2 void TaskRelevanceMapStd::exportOptions (const int *whatToExport*, const bool *recurse* = true) [virtual]

Export some of our [ModelParam](#) parameters as command-line-options.

Reimplemented from [TaskRelevanceMap](#).

Definition at line 198 of file TaskRelevanceMap.C.

References [TaskRelevanceMap::exportOptions\(\)](#), [itsBiasTRM](#), [itsLearnTRM](#), [itsLearnUpdateTRM](#), [OPTEXP\\_CORE](#), and [ModelManager::requestOption\(\)](#).

### 10.295.3.3 int TaskRelevanceMapStd::getHeight () const [virtual]

Get height of the map.

Implements [TaskRelevanceMap](#).

Definition at line 262 of file TaskRelevanceMap.C.

References [Image< T >::getHeight\(\)](#).

Referenced by `update()`.

#### 10.295.3.4 `float TaskRelevanceMapStd::getV (const Point2D & p) const` `[virtual]`

Access function using [Point2D](#).

Implements [TaskRelevanceMap](#).

Definition at line 254 of file `TaskRelevanceMap.C`.

References `Image< float >::getVal()`.

#### 10.295.3.5 `Image< float > TaskRelevanceMapStd::getV (const bool normalize = false) const` `[virtual]`

Return all our relevance values as an [Image](#)<float>.

Implements [TaskRelevanceMap](#).

Definition at line 238 of file `TaskRelevanceMap.C`.

References `Image< T >::clamp()`, and `ModelParam< float >::getVal()`.

#### 10.295.3.6 `int TaskRelevanceMapStd::getWidth () const` `[virtual]`

Get width of the map.

Implements [TaskRelevanceMap](#).

Definition at line 258 of file `TaskRelevanceMap.C`.

References `Image< T >::getWidth()`.

Referenced by `update()`.

#### 10.295.3.7 `void TaskRelevanceMapStd::input (const Image< float > & in)` `[virtual]`

Set new relevance values.

Implements [TaskRelevanceMap](#).

Definition at line 221 of file `TaskRelevanceMap.C`.

References `Image< float >::getDims()`, `itsBiasTRM`, `Raster::ReadGray()`, `rescale()`, and `Image< float >::swap()`.

**10.295.3.8 void TaskRelevanceMapStd::reset () [virtual]**

Reset to initial state just after construction.

Implements [TaskRelevanceMap](#).

Definition at line 217 of file TaskRelevanceMap.C.

References `Image< float >::freeMem()`.

**10.295.3.9 void TaskRelevanceMapStd::update (const [Point2D](#) & fixation, const float relevance) [virtual]**

update TRM with relevance of current fixation

Implements [TaskRelevanceMap](#).

Definition at line 266 of file TaskRelevanceMap.C.

References `SharedPtr< ShapeEstimator >::get()`, `Image< float >::getDims()`, `getHeight()`, `Brain::getInputDims()`, `Image< float >::getMinMax()`, `Brain::getSE()`, `ModelParam< bool >::getVal()`, `Image< float >::getVal()`, `Image< T >::getVal()`, `getWidth()`, `Dims::h()`, `Point2D::i`, `itsLearnTRM`, `Point2D::j`, `Image< float >::normalize()`, `rescale()`, `Image< float >::setVal()`, and `Dims::w()`.

**10.295.4 Member Data Documentation****10.295.4.1 [ModelParam<std::string>](#) TaskRelevanceMapStd::itsBiasTRM [protected]**

If an image is specified with this option, then use it to bias the TRM

Definition at line 262 of file TaskRelevanceMap.H.

Referenced by `exportOptions()`, and `input()`.

**10.295.4.2 [ModelParam<bool>](#) TaskRelevanceMapStd::itsLearnTRM [protected]**

Learn the TRM for the given task by inspecting several scenes

Definition at line 256 of file TaskRelevanceMap.H.

Referenced by `exportOptions()`, and `update()`.



#### 10.295.4.3 [ModelParam<bool>](#) [TaskRelevanceMapStd::itsLearnUpdateTRM](#) [protected]

Learn and update the TRM as we inspecting several scenes

Definition at line 259 of file TaskRelevanceMap.H.

Referenced by [exportOptions\(\)](#).

The documentation for this class was generated from the following files:

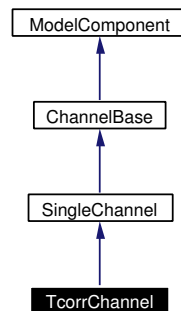
- [TaskRelevanceMap.H](#)
- [TaskRelevanceMap.C](#)

## 10.296 TcorrChannel Class Reference

A temporal correlation channel.

```
#include <Channels.H>
```

Inheritance diagram for TcorrChannel:



Collaboration diagram for TcorrChannel:

### Public Member Functions

- [TcorrChannel](#) ([ModelManager](#) &mgr)  
*Construct with standard params.*
- virtual void [exportOptions](#) (const int whatToExport, const bool recurse=true)  
*Export some of our [ModelParam](#) parameters as command-line-options.*
- virtual [~TcorrChannel](#) ()  
*Destructor.*
- virtual bool [outputAvailable](#) () const  
*overload*

### Protected Member Functions

- virtual void [doInput](#) (const [Image](#)< [PixRGB](#)< float > > \*, const [Image](#)< float > \*bwimg, const double t, const [Image](#)< byte > &clipMask)  
*special input function*

- virtual [Image](#)< float > [getOutput](#) ()  
*special output function*

## Protected Attributes

- [ModelParam](#)< int > [itsFrameLag](#)  
*Lag (in frames) to use for correlation.*
- [ImageCache](#)< float > [itsCache](#)  
*cache of the recent inputs*
- [Image](#)< float > [itsMap](#)  
*our temporal correlation map*

### 10.296.1 Detailed Description

A temporal correlation channel.

Note that in this channel, high 'saliency' means low temporal correlation, i.e., lots of change. CAUTION: this channel will throw a fatal exception if your [LevelSpec](#) contains more than one level or if that level is different from the maplevel or if the deltas are non-zero.

Definition at line 537 of file Channels.H.

### 10.296.2 Constructor & Destructor Documentation

#### 10.296.2.1 TcorrChannel::TcorrChannel ([ModelManager](#) & mgr)

Construct with standard params.

Definition at line 2653 of file Channels.C.

References TCORR.

#### 10.296.2.2 TcorrChannel::~~TcorrChannel () [virtual]

Destructor.

Definition at line 2672 of file Channels.C.

### 10.296.3 Member Function Documentation

**10.296.3.1** `void TcorrChannel::doInput (const Image< PixRGB< float > > *,  
const Image< float > * bwimg, const double t, const Image< byte >  
& clipMask)` [protected, virtual]

special input function

Reimplemented from [SingleChannel](#).

Definition at line 2680 of file Channels.C.

References `ASSERT`, `LevelSpec::delMax()`, `LevelSpec::delMin()`, `ImageCache< float >::getMaxSize()`, `ModelParam< int >::getVal()`, `ModelParam< LevelSpec >::getVal()`, `itsCache`, `itsFrameLag`, `LevelSpec::levMax()`, `LevelSpec::levMin()`, `LevelSpec::mapLevel()`, `ImageCache< float >::push_back()`, and `ImageCache< float >::setMaxSize()`.

**10.296.3.2** `void TcorrChannel::exportOptions (const int whatToExport, const  
bool recurse = true)` [virtual]

Export some of our [ModelParam](#) parameters as command-line-options.

See the base function in [ModelComponent.H](#) for info.

Reimplemented from [SingleChannel](#).

Definition at line 2661 of file Channels.C.

References `SingleChannel::exportOptions()`, `itsFrameLag`, `OPTEXP_CORE`, and `ModelManager::requestOption()`.

**10.296.3.3** `Image< float > TcorrChannel::getOutput ()` [protected,  
virtual]

special output function

Reimplemented from [SingleChannel](#).

Definition at line 2700 of file Channels.C.

References `ImageCache< float >::back()`, `Image< float >::beginw()`, `corrpatch()`, `ImageCache< float >::front()`, `Image< T >::getHeight()`, `ImageCache< float >::getMaxSize()`, `ModelParam< LevelSpec >::getVal()`, `Image< T >::getWidth()`, `itsCache`, `LevelSpec::mapLevel()`, `Image< float >::resize()`, `ImageCache< float >::size()`, and `uint`.

**10.296.3.4 bool TcorrChannel::outputAvailable () const [virtual]**

overload

Reimplemented from [SingleChannel](#).

Definition at line 2676 of file Channels.C.

References [ImageCache< float >::getMaxSize\(\)](#), [itsCache](#), and [ImageCache< float >::size\(\)](#).

**10.296.4 Member Data Documentation****10.296.4.1 [ImageCache<float>](#) TcorrChannel::itsCache [protected]**

cache of the recent inputs

Definition at line 565 of file Channels.H.

Referenced by [doInput\(\)](#), [getOutput\(\)](#), and [outputAvailable\(\)](#).

**10.296.4.2 [ModelParam<int>](#) TcorrChannel::itsFrameLag [protected]**

Lag (in frames) to use for correlation.

Definition at line 555 of file Channels.H.

Referenced by [doInput\(\)](#), and [exportOptions\(\)](#).

**10.296.4.3 [Image<float>](#) TcorrChannel::itsMap [protected]**

our temporal correlation map

Definition at line 566 of file Channels.H.

The documentation for this class was generated from the following files:

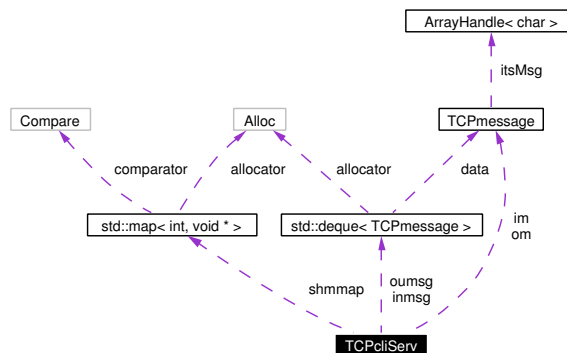
- [Channels.H](#)
- [Channels.C](#)

## 10.297 TCPcliServ Class Reference

A client/server to receive/send [TCPmessage](#).

```
#include <TCPcliServ.H>
```

Collaboration diagram for TCPcliServ:



### Public Member Functions

- [TCPcliServ](#) ()

Base constructor. Must call [init\(\)](#) later.

- void [init](#) (const int connected\_fd, const in\_addr\_t my\_ipaddr, const short int my\_port, const in\_addr\_t cli\_ipaddr, const short int cli\_port, const int inqlen=0, const int outqlen=0, const bool indlast=false, const bool outdlast=false, const bool disableShm=false)

(re)-initialization from a connected socket

- [TCPcliServ](#) (const int connected\_fd, const in\_addr\_t my\_ipaddr, const short int my\_port, const in\_addr\_t cli\_ipaddr, const short int cli\_port, const int inqlen=0, const int outqlen=0, const bool indlast=false, const bool outdlast=false, const bool disableShm=false)

Constructor and initialization from a connected socket.

- [~TCPcliServ](#) ()

Destructor.

- void [send](#) (const [TCPmessage](#) &msg)

Send a message.

- bool [receive](#) (TCPmessage &msg)  
*See if we have received a message (returns false if no new message).*
- int [nbReceived](#) ()  
*Do we have any received messages in our queue.*
- int [check](#) ()  
*Check for new incoming message and do transfers.*
- int [disconnect](#) ()  
*Terminate the connection.*

### 10.297.1 Detailed Description

A client/server to receive/send [TCPmessage](#).

This class defines a simple client/server that handles one TCP connection and can send/receive TCPmessages over it. Messages to be sent or being received are queued; the actual transmission happens in [check\(\)](#) which should be called often, typically under the control of a [TCPcommunicator](#) object that uses a [SockServ](#) object to know when our TCPcliServ needs attention.

this class uses a mutex to allow [check\(\)](#) to be called from one thread while [send\(\)](#) and [receive\(\)](#) may simultaneously be called from another thread.

Definition at line 60 of file TCPcliServ.H.

### 10.297.2 Constructor & Destructor Documentation

#### 10.297.2.1 TCPcliServ::TCPcliServ ()

Base constructor. Must call [init\(\)](#) later.

Definition at line 64 of file TCPcliServ.C.

#### 10.297.2.2 TCPcliServ::TCPcliServ (const int *connected\_fd*, const in\_addr\_t *my\_ipaddr*, const short int *my\_port*, const in\_addr\_t *cli\_ipaddr*, const short int *cli\_port*, const int *inqlen* = 0, const int *outqlen* = 0, const bool *indlast* = false, const bool *outdlast* = false, const bool *disableShm* = false)

Constructor and initialization from a connected socket.

Definition at line 115 of file TCPcliServ.C.

References `init()`.

### 10.297.2.3 `TCPcliServ::~~TCPcliServ ()`

Destructor.

Definition at line 129 of file `TCPcliServ.C`.

References `disconnect()`.

## 10.297.3 Member Function Documentation

### 10.297.3.1 `int TCPcliServ::check ()`

Check for new incoming message and do transfers.

Check for new incoming message, continue reading/writing current messages, or start writing out a new message from the write queue (thread-safe). This must be called often for data to flow.

Definition at line 189 of file `TCPcliServ.C`.

References `TCPmessage::addShmInfo()`, `DONE`, `TCPmessage::getAction()`, `TCPmessage::getID()`, `TCPmessage::getSize()`, `TCPmessage::readFrom()`, `TCPmessage::readHeaderFrom()`, `TCPmessage::reset()`, `TCPmessage::writeHeaderTo()`, and `TCPmessage::writeTo()`.

Referenced by `TCPcommunicator::run()`.

### 10.297.3.2 `int TCPcliServ::disconnect ()`

Terminate the connection.

Definition at line 323 of file `TCPcliServ.C`.

Referenced by `~TCPcliServ()`.

### 10.297.3.3 `void TCPcliServ::init (const int connected_fd, const in_addr_t my_ipaddr, const short int my_port, const in_addr_t cli_ipaddr, const short int cli_port, const int inqlen = 0, const int outqlen = 0, const bool indlast = false, const bool outdlast = false, const bool disableShm = false)`

(re)-initialization from a connected socket

`my_ipaddr` and `my_port` should be my IP address and port. `cli_ipaddr` and `cli_port` should be the IP address and port of the client on the other end of `connected_fd`. For



now, they are only used to determine whether we can optimize transfers by using shared memory in case the client's IP address matches ours. inqlen and outqlen are the maximum incoming and outgoing queue length, or zero for unlimited. indlast and outdlast specify the drop policies for the input and output queues; if true, the last message (most recent) coming into the queue will be dropped if the queue is full, otherwise the oldest message already in the queue will be dropped.

Definition at line 68 of file TCPcliServ.C.

Referenced by TCPcliServ().

#### 10.297.3.4 int TCPcliServ::nbReceived ()

Do we have any received messages in our queue.

Returns the number of messages that have been fully received (not those in progress) and are waiting in the queue. Use [receive\(\)](#) to get the messages out of the incoming queue.

Definition at line 179 of file TCPcliServ.C.

Referenced by TCPcommunicator::nbReceived().

#### 10.297.3.5 bool TCPcliServ::receive ([TCPmessage](#) & msg)

See if we have received a message (returns false if no new message).

Definition at line 168 of file TCPcliServ.C.

Referenced by TCPcommunicator::receiveIt().

#### 10.297.3.6 void TCPcliServ::send (const [TCPmessage](#) & msg)

Send a message.

Definition at line 141 of file TCPcliServ.C.

References [TCPmessage::checkShmDone\(\)](#).

Referenced by TCPcommunicator::send().

The documentation for this class was generated from the following files:

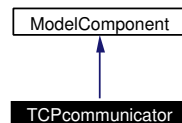
- [TCPcliServ.H](#)
- [TCPcliServ.C](#)

## 10.298 TCPcommunicator Class Reference

This class handles passing TCPmessages in and out.

```
#include <TCPcommunicator.H>
```

Inheritance diagram for TCPcommunicator:



Collaboration diagram for TCPcommunicator:

### Public Member Functions

- [TCPcommunicator](#) ([ModelManager](#) &mgr, const [std::string](#) &descrName="TCP Communicator", const [std::string](#) &tagName="TCPcommunicator")  
*default constructor*
- virtual void [exportOptions](#) (const int whatToExport, const bool recurse=true)  
*Export some of our [ModelParam](#) parameters as command-line-options.*
- virtual [~TCPcommunicator](#) ()  
*TCPcommunicator destructor.*
- int [contact](#) (const char \*hostname, const bool [blocking](#)=true)  
*contact a peer (return fd). hostname contains name:port*
- int [contact](#) (const in\_addr\_t ip, const short int port, const bool [blocking](#)=true)  
*contact a peer using its IP and port (returns fd)*
- void [send](#) (const int sfd, [TCPmessage](#) &smsg)  
*send a message to a known peer:*
- bool [receive](#) (int &rfd, [TCPmessage](#) &rmsg, const int timeout=0)  
*receive a message*
- int [nbReceived](#) (const int rfd=-1)  
*Do we have any received messages?*

- void [waitFor](#) (const int rfd, const int timeout)  
*Wait for availability of an incoming message.*
- void [terminateAll](#) ()  
*terminate all open connections (except master server for new incoming)*
- void [terminateAllButOne](#) (const int fd)  
*terminate all open connections but one (and master server)*
- void [run](#) ()  
*thread for data transfers*

### Protected Member Functions

- void [start1](#) ()  
*get started and initialize ourselves (before [SockServ](#) start)*
- void [start2](#) ()  
*get started and initialize ourselves (after [SockServ](#) start)*
- void [stop1](#) ()  
*get stopped (before [SockServ](#) stop)*
- void [pollvar](#) (volatile bool \*var)  
*poll var until it becomes false*
- bool [receiveIt](#) (int &rfd, [TCPmessage](#) &rmsg)  
*internal non-blocking receive check*
- void [terminateCli](#) (const int fd)  
*terminate a [TCPcliServ](#)*

### Protected Attributes

- [ModelParam](#)< in\_addr > [myIP](#)  
*My IP address to listen on.*
- [ModelParam](#)< int > [inqlen](#)  
*incoming queue lengths*

- [ModelParam](#)< int > [ouqlen](#)  
*outgoing queue lengths*
- [ModelParam](#)< bool > [indroplast](#)  
*incoming queue drop policy*
- [ModelParam](#)< bool > [oudroplast](#)  
*outgoing queue drop policy*
- [ModelParam](#)< bool > [disableShm](#)  
*disbale shm transfers if true*
- [SharedPtr](#)< [SockServ](#) > [server](#)  
*manages incoming connections and transfers*
- [TCPcliServ](#) \*\* [cli](#)  
*array of monitored TCPcliServ's*
- pthread\_t [runner](#)  
*all transfers are done via a separate thread*
- bool [to\\_add](#)  
*these are for communication between main and running threads:*
- bool [to\\_stop](#)  
*these are for communication between main and running threads:*
- int **addfd**
- in\_addr\_t **addip**
- short int **addport**

### 10.298.1 Detailed Description

This class handles passing TCPmessages in and out.

This class handles passing TCPmessages in and out; CAUTION: it is a multi-threaded class. Users can establish a connection with another host also running a TCPcommunicator by using [contact\(\)](#). Then, they can send out TCPmessages to that host using [send\(\)](#). Similarly, other hosts may contact us and send us messages, which we can then retrieve using [receive\(\)](#).

Definition at line 61 of file TCPcommunicator.H.

## 10.298.2 Constructor & Destructor Documentation

### 10.298.2.1 TCPcommunicator::TCPcommunicator ([ModelManager](#) & mgr, const [std::string](#) & descrName = "TCP Communicator", const [std::string](#) & tagName = "TCPcommunicator")

default constructor

Definition at line 69 of file TCPcommunicator.C.

References [ModelComponent::addSubComponent\(\)](#), and [server](#).

### 10.298.2.2 TCPcommunicator::~TCPcommunicator () [virtual]

TCPcommunicator destructor.

Definition at line 148 of file TCPcommunicator.C.

## 10.298.3 Member Function Documentation

### 10.298.3.1 int TCPcommunicator::contact (const [in\\_addr\\_t](#) ip, const short int port, const bool blocking = true)

contact a peer using its IP and port (returns fd)

Definition at line 172 of file TCPcommunicator.C.

References [ASSERT](#), [pollvar\(\)](#), [ModelComponent::started\(\)](#), and [to\\_add](#).

### 10.298.3.2 int TCPcommunicator::contact (const char \* hostname, const bool blocking = true)

contact a peer (return fd). hostname contains name:port

Definition at line 152 of file TCPcommunicator.C.

References [DEFPORT](#).

### 10.298.3.3 void TCPcommunicator::exportOptions (const int whatToExport, const bool recurse = true) [virtual]

Export some of our [ModelParam](#) parameters as command-line-options.

See the base function in [ModelComponent.H](#) for info.

Reimplemented from [ModelComponent](#).

Definition at line 86 of file TCPcommunicator.C.

References `disableShm`, `ModelComponent::exportOptions()`, `inqlen`, `OPTEXP_CORE`, `oudroplast`, `ouqlen`, and `ModelManager::requestOption()`.

#### 10.298.3.4 `int TCPcommunicator::nbReceived (const int rfd = -1)`

Do we have any received messages?

Returns the total number of messages in the incoming queues of our various [TCPcli-Serv](#) members. If `rfd != -1`, only consider a given client (`fd`) rather than any client.

Definition at line 364 of file `TCPcommunicator.C`.

References `TCPcliServ::nbReceived()`.

#### 10.298.3.5 `void TCPcommunicator::pollvar (volatile bool * var) [protected]`

poll var until it becomes false

Definition at line 396 of file `TCPcommunicator.C`.

References `POLL_SLEEP`.

Referenced by `contact()`, and `stop1()`.

#### 10.298.3.6 `bool TCPcommunicator::receive (int & rfd, TCPmessage & rmsg, const int timeout = 0)`

receive a message

receive a message (returns false if no new message); use `rfd = -1` to receive from any currently connected `fd` (`fd` received from will then be put in `rfd`). If `timeout` is zero, the call returns immediately, otherwise may wait up to `timeout` ms.

Definition at line 341 of file `TCPcommunicator.C`.

References `receiveIt()`, and `waitFor()`.

#### 10.298.3.7 `bool TCPcommunicator::receiveIt (int & rfd, TCPmessage & rmsg) [protected]`

internal non-blocking receive check

Definition at line 407 of file `TCPcommunicator.C`.

References `ASSERT`, and `TCPcliServ::receive()`.

Referenced by `receive()`.

### 10.298.3.8 void TCPcommunicator::run ()

thread for data transfers

this will run in a different thread and take care of the actual data transfers, off-loading the main thread from them. Do not call this directly!

Definition at line 208 of file TCPcommunicator.C.

References TCPcliServ::check(), disableShm, DONE, ModelParam< bool >::getVal(), ModelParam< int >::getVal(), ModelParam< in\_addr >::getVal(), inqlen, oudroplast, ouqlen, server, SOCKSERV\_IDLE, terminateCli(), and to\_add.

### 10.298.3.9 void TCPcommunicator::send (const int *sfd*, TCPmessage & *msg*)

send a message to a known peer:

Definition at line 327 of file TCPcommunicator.C.

References ASSERT, TCPcliServ::send(), and server.

### 10.298.3.10 void TCPcommunicator::start1 () [protected, virtual]

get started and initialize ourselves (before [SockServ](#) start)

Reimplemented from [ModelComponent](#).

Definition at line 103 of file TCPcommunicator.C.

References ModelParam< in\_addr >::getVal(), ModelParam< in\_addr >::getValString(), and ModelParam< in\_addr >::setVal().

### 10.298.3.11 void TCPcommunicator::start2 () [protected, virtual]

get started and initialize ourselves (after [SockServ](#) start)

Reimplemented from [ModelComponent](#).

Definition at line 128 of file TCPcommunicator.C.

References to\_add.

### 10.298.3.12 void TCPcommunicator::stop1 () [protected, virtual]

get stopped (before [SockServ](#) stop)

Reimplemented from [ModelComponent](#).

Definition at line 141 of file TCPcommunicator.C.

References pollvar().

#### **10.298.3.13 void TCPcommunicator::terminateAll ()**

terminate all open connections (except master server for new incoming)

Definition at line 307 of file TCPcommunicator.C.

References terminateCli().

#### **10.298.3.14 void TCPcommunicator::terminateAllButOne (const int *fd*)**

terminate all open connections but one (and master server)

Definition at line 311 of file TCPcommunicator.C.

References terminateCli().

#### **10.298.3.15 void TCPcommunicator::terminateCli (const int *fd*) [protected]**

terminate a [TCPcliServ](#)

Definition at line 315 of file TCPcommunicator.C.

References ASSERT, and server.

Referenced by run(), terminateAll(), and terminateAllButOne().

#### **10.298.3.16 void TCPcommunicator::waitFor (const int *rfd*, const int *timeout*)**

Wait for availability of an incoming message.

use rfd = -1 to wait for message on any currently connected fd; timeout is in milliseconds.

Definition at line 381 of file TCPcommunicator.C.

Referenced by receive().

### **10.298.4 Member Data Documentation**

#### **10.298.4.1 [TCPcliServ](#)\*\* [TCPcommunicator::cli](#) [protected]**

array of monitored TCPcliServ's

Definition at line 144 of file TCPcommunicator.H.



**10.298.4.2   [ModelParam<bool> TCPcommunicator::disableShm](#) [protected]**

disbale shm transfers if true

Definition at line 122 of file TCPcommunicator.H.

Referenced by exportOptions(), and run().

**10.298.4.3   [ModelParam<bool> TCPcommunicator::indroplast](#) [protected]**

incoming queue drop policy

Definition at line 120 of file TCPcommunicator.H.

**10.298.4.4   [ModelParam<int> TCPcommunicator::inqlen](#) [protected]**

incoming queue lengths

Definition at line 118 of file TCPcommunicator.H.

Referenced by exportOptions(), and run().

**10.298.4.5   [ModelParam<in\\_addr> TCPcommunicator::myIP](#) [protected]**

My IP address to listen on.

Definition at line 117 of file TCPcommunicator.H.

**10.298.4.6   [ModelParam<bool> TCPcommunicator::oudroplast](#) [protected]**

outgoing queue drop policy

Definition at line 121 of file TCPcommunicator.H.

Referenced by exportOptions(), and run().

**10.298.4.7   [ModelParam<int> TCPcommunicator::ouqlen](#) [protected]**

outgoing queue lengths

Definition at line 119 of file TCPcommunicator.H.

Referenced by exportOptions(), and run().

**10.298.4.8** `pthread_t TCPcommunicator::runner` [protected]

all transfers are done via a separate thread

Definition at line 147 of file TCPcommunicator.H.

**10.298.4.9** `SharedPtr<SockServ> TCPcommunicator::server` [protected]

manages incoming connections and transfers

Definition at line 125 of file TCPcommunicator.H.

Referenced by `run()`, `send()`, `TCPcommunicator()`, and `terminateCli()`.

**10.298.4.10** `bool TCPcommunicator::to_add` [mutable, protected]

these are for communication between main and running threads:

Definition at line 150 of file TCPcommunicator.H.

Referenced by `contact()`, `run()`, and `start2()`.

**10.298.4.11** `bool TCPcommunicator::to_stop` [mutable, protected]

these are for communication between main and running threads:

Definition at line 150 of file TCPcommunicator.H.

The documentation for this class was generated from the following files:

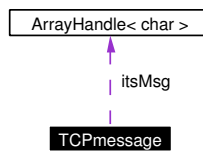
- [TCPcommunicator.H](#)
- [TCPcommunicator.C](#)

## 10.299 TCPmessage Class Reference

This class defines a modular message to be passed over TCP connections.

```
#include <TCPmessage.H>
```

Collaboration diagram for TCPmessage:



### Public Member Functions

- [TCPmessage \(\)](#)  
*uninitialized constructor (for pointers)*
- [TCPmessage \(const int32 msgid, const int32 msgaction, const float msgeti=0.0F\)](#)  
*constructor with empty message body*
- [TCPmessage \(const int32 msgid, const int32 msgaction, const float msgeti, const void \\*buf, const int bufsize\)](#)  
*constructor with copy of given message body*
- [TCPmessage \(const TCPmessage &m\)](#)  
*copy constructor*
- [TCPmessage & operator= \(const TCPmessage &m\)](#)  
*assignment*
- [~TCPmessage \(\)](#)  
*destructor*
- void [addImage \(const Image< PixRGB< byte > > &im\)](#)  
*method to add elements into a message to be sent out*
- void [addImage \(const Image< byte > &im\)](#)  
*method to add elements into a message to be sent out*

- void [addImage](#) (const [Image](#)< float > &im)  
*method to add elements into a message to be sent out*
- void [addImageSet](#) (const [ImageSet](#)< float > &im)  
*method to add elements into a message to be sent out*
- void [addFixation](#) (const [int32](#) i, const [int32](#) j, const [int32](#) frame)  
*method to add elements into a message to be sent out*
- void [addString](#) (const char \*str)  
*method to add elements into a message to be sent out*
- void [addInt32](#) (const [int32](#) val)  
*method to add elements into a message to be sent out*
- void [addFloat](#) (const float val)  
*method to add elements into a message to be sent out*
- void [addDouble](#) (const double val)  
*method to add elements into a message to be sent out*
- void [setID](#) (const [int32](#) id)  
*modify message ID*
- [int32](#) [getID](#) () const  
*get message ID*
- void [setAction](#) (const [int32](#) ac)  
*modify message action*
- [int32](#) [getAction](#) () const  
*get message action*
- void [setETI](#) (const float eti)  
*modify message eti (estimated time to idle)*
- float [getETI](#) () const  
*get message eti (estimated time to idle)*
- [int32](#) [getSize](#) () const  
*get total message size (mostly for debugging purposes)*

- bool [isBusy](#) () const  
*check whether message is busy (sending or receiving data):*
- void [reset](#) (const [int32](#) msgid, const [int32](#) msgaction, const float msgeti=0.0F)  
*empty old contents and get ready for a re-use:*
- void [freeMem](#) ()  
*free memory:*
- void [getElement](#) (void \*\*elem, [int32](#) &typ)  
*methods to get elements out of a received message*
- template<class T> void [unpack](#) (const T \*data, const int nitem)  
*unpack some data from message (memory must be pre-allocated):*

## Friends

- class [TCPcliServ](#)  
*make our private methods accessible to [TCPcliServ](#):*

### 10.299.1 Detailed Description

This class defines a modular message to be passed over TCP connections.

The basic message structure is as follows:

- one TCPmessageHeader (see private section of TCPmessage definition)
- [repeat as needed:]
- field type ([int32](#))
- field data (size and type depending on field type)
- field type = TCPMSG\_END CAUTION: to optimize speed, there is no semaphore when modifying the contents of the message (e.g., adding fields) or when parsing it. So, be careful with multi-threaded uses!

Definition at line 78 of file TCPmessage.H.

## 10.299.2 Constructor & Destructor Documentation

### 10.299.2.1 TCPmessage::TCPmessage ()

uninitialized constructor (for pointers)

Definition at line 54 of file TCPmessage.C.

References `reset()`.

### 10.299.2.2 TCPmessage::TCPmessage (const [int32](#) *msgid*, const [int32](#) *msgaction*, const float *msgeti* = 0.0F)

constructor with empty message body

#### Parameters:

***msgid*** a message identification (user-defined)

***msgaction*** additional field where users can specify an action to be executed when this message is received

***msgeti*** estimated time to idle, in seconds. This is used for automatic load balancing, where the [Beowulf](#) class will dispatch processing to nodes with shorter *eti*'s in priority, when a load-balanced send is requested. If you are not doing load balancing, just use the default of 0.0.

Definition at line 59 of file TCPmessage.C.

References `reset()`.

### 10.299.2.3 TCPmessage::TCPmessage (const [int32](#) *msgid*, const [int32](#) *msgaction*, const float *msgeti*, const void \* *buf*, const int *bufsize*)

constructor with copy of given message body

NOTE: we do a deep copy of the given buffer, so that it can be freed immediately upon return from this constructor

Definition at line 65 of file TCPmessage.C.

References `reset()`.

### 10.299.2.4 TCPmessage::TCPmessage (const [TCPmessage](#) & *m*)

copy constructor

Definition at line 77 of file TCPmessage.C.

References `itsBusy`, `itsHead`, and `itsUpkidx`.

### 10.299.2.5 TCPmessage::~~TCPmessage ()

destructor

Definition at line 90 of file TCPmessage.C.

## 10.299.3 Member Function Documentation

### 10.299.3.1 void TCPmessage::addDouble (const double val)

method to add elements into a message to be sent out

Definition at line 154 of file TCPmessage.C.

Referenced by SingleChannelBeoServer::check(), and SingleChannelBeo::send-Input().

### 10.299.3.2 void TCPmessage::addFixation (const int32 i, const int32 j, const int32 frame)

method to add elements into a message to be sent out

Definition at line 129 of file TCPmessage.C.

### 10.299.3.3 void TCPmessage::addFloat (const float val)

method to add elements into a message to be sent out

Definition at line 148 of file TCPmessage.C.

### 10.299.3.4 void TCPmessage::addImage (const Image< float > & im)

method to add elements into a message to be sent out

Definition at line 112 of file TCPmessage.C.

References Image< T >::getArrayPtr(), Image< T >::getHeight(), Image< T >::getSize(), and Image< T >::getWidth().

### 10.299.3.5 void TCPmessage::addImage (const Image< byte > & im)

method to add elements into a message to be sent out

Definition at line 103 of file TCPmessage.C.

References Image< T >::getArrayPtr(), Image< T >::getHeight(), Image< T >::getSize(), and Image< T >::getWidth().

**10.299.3.6 void TCPmessage::addImage (const Image< PixRGB< byte > > & im)**

method to add elements into a message to be sent out

Definition at line 94 of file TCPmessage.C.

Referenced by addImageSet(), SingleChannelBeoServer::check(), computeCMAP(), main(), BeobotVisualCortex::masterProcess(), and SingleChannelBeo::sendInput().

**10.299.3.7 void TCPmessage::addImageSet (const ImageSet< float > & im)**

method to add elements into a message to be sent out

Definition at line 121 of file TCPmessage.C.

References addImage(), ImageSet< T >::size(), and uint.

Referenced by SingleChannelBeoServer::check().

**10.299.3.8 void TCPmessage::addInt32 (const int32 val)**

method to add elements into a message to be sent out

Definition at line 142 of file TCPmessage.C.

Referenced by main().

**10.299.3.9 void TCPmessage::addString (const char \* str)**

method to add elements into a message to be sent out

Definition at line 136 of file TCPmessage.C.

Referenced by SingleChannelBeo::sendInput().

**10.299.3.10 void TCPmessage::freeMem ()**

free memory:

Definition at line 169 of file TCPmessage.C.

References ArrayHandle< char >::swap().

Referenced by reset().



**10.299.3.11** [int32](#) TCPmessage::getAction () const [inline]

get message action

Definition at line 253 of file TCPmessage.H.

Referenced by TCPcliServ::check(), Beowulf::receive(), and Beowulf::send().

**10.299.3.12** void TCPmessage::getElement (void \*\* *elem*, [int32](#) & *typ*)

methods to get elements out of a received message

methods to get elements out of a received message (*typ* will contain TCPMSG\_XXX so that the type of *elem* can be inferred). CAUTION: object will be allocated with new; you must delete it when done with it!

Definition at line 180 of file TCPmessage.C.

References Image< T >::getArrayPtr(), NO\_INIT, and unpack().

Referenced by SingleChannelBeoServer::check(), computeCMap(), computeCMap2(), BeoSub::grab(), main(), BeobotVisualCortex::masterCollect(), receiveCMaps(), BeobotVisualCortex::slaveProcess(), and SingleChannelBeo::waitForOutput().

**10.299.3.13** float TCPmessage::getETI () const [inline]

get message eti (estimated time to idle)

Definition at line 261 of file TCPmessage.H.

Referenced by Beowulf::receive().

**10.299.3.14** [int32](#) TCPmessage::getID () const [inline]

get message ID

Definition at line 245 of file TCPmessage.H.

Referenced by TCPcliServ::check(), computeCMap(), computeCMap2(), Beowulf::receive(), and Beowulf::send().

**10.299.3.15** [int32](#) TCPmessage::getSize () const [inline]

get total message size (mostly for debugging purposes)

Definition at line 265 of file TCPmessage.H.

Referenced by TCPcliServ::check().

**10.299.3.16** `bool TCPmessage::isBusy () const [inline]`

check whether message is busy (sending or receiving data):

Definition at line 269 of file TCPmessage.H.

**10.299.3.17** `TCPmessage & TCPmessage::operator= (const TCPmessage & m)`

assignment

Definition at line 82 of file TCPmessage.C.

References itsBusy, itsHead, itsMsg, itsUpkidx, and ArrayHandle< char >::swap().

**10.299.3.18** `void TCPmessage::reset (const int32 msgid, const int32 msgaction, const float msgeti = 0.0F)`

empty old contents and get ready for a re-use:

Definition at line 160 of file TCPmessage.C.

References freeMem().

Referenced by TCPcliServ::check(), SingleChannelBeoServer::check(), main(), BeobotVisualCortex::masterProcess(), and TCPmessage().

**10.299.3.19** `void TCPmessage::setAction (const int32 ac) [inline]`

modify message action

Definition at line 249 of file TCPmessage.H.

Referenced by BeobotVisualCortex::masterProcess().

**10.299.3.20** `void TCPmessage::setETI (const float eti) [inline]`

modify message eti (estimated time to idle)

Definition at line 257 of file TCPmessage.H.

**10.299.3.21** `void TCPmessage::setID (const int32 id) [inline]`

modify message ID

Definition at line 241 of file TCPmessage.H.

**10.299.3.22** `template<class T> void TCPmessage::unpack (const T * data,  
const int nitem) [inline]`

unpack some data from message (memory must be pre-allocated):

Definition at line 307 of file TCPmessage.H.

Referenced by getElement().

## 10.299.4 Friends And Related Function Documentation

**10.299.4.1** `friend class TCPcliServ [friend]`

make our private methods accessible to [TCPcliServ](#):

Definition at line 200 of file TCPmessage.H.

The documentation for this class was generated from the following files:

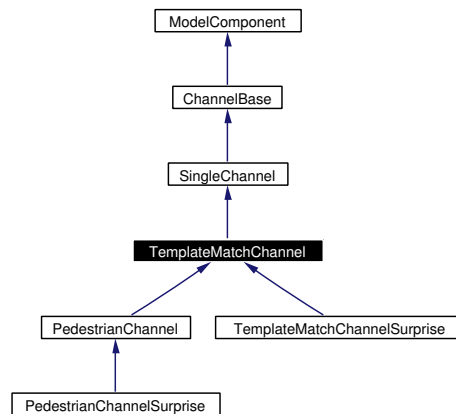
- [TCPmessage.H](#)
- [TCPmessage.C](#)

## 10.300 TemplateMatchChannel Class Reference

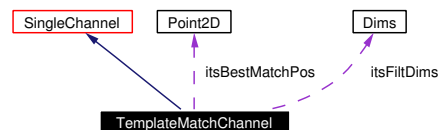
Convolves each level of its pyramid with a feature template.

```
#include <Channels.H>
```

Inheritance diagram for TemplateMatchChannel:



Collaboration diagram for TemplateMatchChannel:



### Public Member Functions

- **TemplateMatchChannel** (**ModelManager** &mgr, const **Image**< float > &filt)  
*Construct with the standard params, plus a filter specification.*
- virtual **~TemplateMatchChannel** ()  
*Virtual destructor.*
- void **findBestMatch** (const **Rectangle** &r, **Point2D** &matchpos, int &match\_lev, float &score)  
*Look for the most likely template match in a given rectangular region.*

- virtual void [drawResults](#) ([Image](#)< [PixRGB](#)< [byte](#) > > &traj)  
*Draws a box in traj around the last best template match found.*

## Protected Member Functions

- virtual void [start1](#) ()

## Protected Attributes

- [Point2D](#) [itsBestMatchPos](#)
- int [itsBestMatchLev](#)
- float [itsBestMatchScore](#)
- [Dims](#) [itsFiltDims](#)

### 10.300.1 Detailed Description

Convolve each level of its pyramid with a feature template.

This type of channel is useful for searching for a specific feature shape at multiple scales. For example, it is used by [VisualCortex](#) to search for possible pedestrians in an input image.

Definition at line 370 of file Channels.H.

### 10.300.2 Constructor & Destructor Documentation

#### 10.300.2.1 [TemplateMatchChannel::TemplateMatchChannel](#) ([ModelManager](#) & mgr, const [Image](#)< float > & filt)

Construct with the standard params, plus a filter specification.

The filter is specified by a C array, which will be interpreted to have dimensions *filt\_x* and *filt\_y*. A private copy of the filter will be made, so the *filt* pointer does not have to be persistent.

Definition at line 1559 of file Channels.C.

References [makeShared\(\)](#), and [TEMPLATE](#).

#### 10.300.2.2 [TemplateMatchChannel::~~TemplateMatchChannel](#) () [virtual]

Virtual destructor.

Definition at line 1570 of file Channels.C.

### 10.300.3 Member Function Documentation

#### 10.300.3.1 `void TemplateMatchChannel::drawResults (Image< PixRGB< byte > > & traj) [virtual]`

Draws a box in *traj* around the last best template match found.

Reimplemented from [ChannelBase](#).

Definition at line 1610 of file Channels.C.

References [Dims::h\(\)](#), [Point2D::i](#), [Point2D::j](#), and [Dims::w\(\)](#).

#### 10.300.3.2 `void TemplateMatchChannel::findBestMatch (const Rectangle & r, Point2D & matchpos, int & match_lev, float & score)`

Look for the most likely template match in a given rectangular region.

The position of the match in 0-level coordinates is returned in *matchpos*, the pyramid level at which the match was found is returned in *match\_lev*, and a score rating the goodness of the match is returned in *score*.

Definition at line 1579 of file Channels.C.

References [crop\(\)](#), [Image< T >::findMax\(\)](#), [SingleChannel::getInputDims\(\)](#), [ModelParam< LevelSpec >::getVal\(\)](#), [Point2D::i](#), [Point2D::j](#), [Rectangle::left\(\)](#), [LevelSpec::levMax\(\)](#), [LevelSpec::levMin\(\)](#), [rescale\(\)](#), [Rectangle::top\(\)](#), and [uint](#).

#### 10.300.3.3 `void TemplateMatchChannel::start1 () [protected, virtual]`

If you overload this, make sure you call [SingleChannel::start1\(\)](#) at the beginning of your overload

Reimplemented from [SingleChannel](#).

Reimplemented in [PedestrianChannel](#).

Definition at line 1573 of file Channels.C.

References [SingleChannel::start1\(\)](#).

Referenced by [PedestrianChannel::start1\(\)](#).

The documentation for this class was generated from the following files:

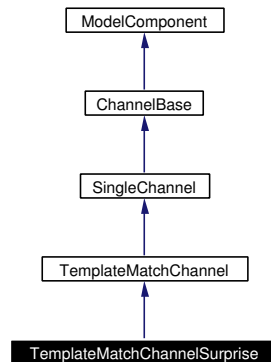
- [Channels.H](#)
- [Channels.C](#)

## 10.301 TemplateMatchChannelSurprise Class Reference

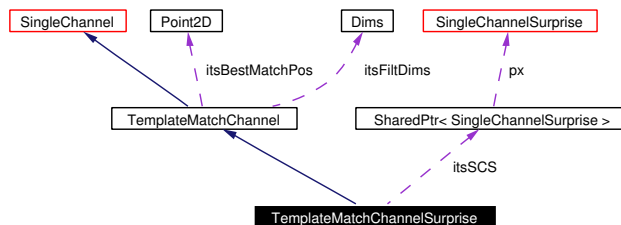
Convolves each level of its pyramid with a feature template.

```
#include <ChannelsSurprise.H>
```

Inheritance diagram for TemplateMatchChannelSurprise:



Collaboration diagram for TemplateMatchChannelSurprise:



### Public Member Functions

- **TemplateMatchChannelSurprise** (**ModelManager** &mgr, const **Image**< float > &filt)  
Construct with the standard params, plus a filter specification.
- virtual **~TemplateMatchChannelSurprise** ()  
Virtual destructor.

- virtual `Image< float > computeSubmap` (const `uint` index) const  
*`computeSubmap()` is special and will return surprise*

## Protected Attributes

- `SharedPtr< SingleChannelSurprise > itsSCS`  
*our surprise computer*

### 10.301.1 Detailed Description

Convolve each level of its pyramid with a feature template.

Definition at line 341 of file ChannelsSurprise.H.

### 10.301.2 Constructor & Destructor Documentation

#### 10.301.2.1 `TemplateMatchChannelSurprise::TemplateMatchChannelSurprise` (`ModelManager` & *mgr*, const `Image< float > & filt`)

Construct with the standard params, plus a filter specification.

Definition at line 311 of file ChannelsSurprise.C.

References `ModelComponent::addSubComponent()`.

#### 10.301.2.2 `TemplateMatchChannelSurprise::~~TemplateMatchChannelSurprise()` [virtual]

Virtual destructor.

Definition at line 317 of file ChannelsSurprise.C.

### 10.301.3 Member Function Documentation

#### 10.301.3.1 `Image< float > TemplateMatchChannelSurprise::computeSubmap` (const `uint` index) const [virtual]

`computeSubmap()` is special and will return surprise

Reimplemented from `SingleChannel`.

Definition at line 321 of file ChannelsSurprise.C.



References uint.

#### 10.301.4 Member Data Documentation

##### 10.301.4.1 [SharedPtr<SingleChannelSurprise>](#) [TemplateMatchChannelSurprise::itsSCS](#) [protected]

our surprise computer

Definition at line 354 of file ChannelsSurprise.H.

The documentation for this class was generated from the following files:

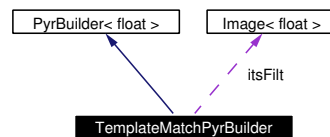
- [ChannelsSurprise.H](#)
- ChannelsSurprise.C

## 10.302 TemplateMatchPyrBuilder Class Reference

A [PyrBuilder](#) that does template matching.

```
#include <PyrBuilder.H>
```

Inheritance diagram for TemplateMatchPyrBuilder: Collaboration diagram for TemplateMatchPyrBuilder:



### Public Member Functions

- **TemplateMatchPyrBuilder** (const [Image](#)< float > &templ)
- virtual [TemplateMatchPyrBuilder](#) \* **clone** () const  
*Cloning constructor useful to make clones from pointers:.*
- virtual [ImageSet](#)< float > **build** (const [Image](#)< float > &image, const int depth)  
*Create a pyramid from an input image.*

### 10.302.1 Detailed Description

A [PyrBuilder](#) that does template matching.

Definition at line 294 of file PyrBuilder.H.

### 10.302.2 Member Function Documentation

#### 10.302.2.1 [ImageSet](#)< float > TemplateMatchPyrBuilder::build (const [Image](#)< float > &image, const int depth) [virtual]

Create a pyramid from an input image.

Implements [PyrBuilder](#)< float >.

Definition at line 300 of file PyrBuilder.C.

References [decX\(\)](#), [decY\(\)](#), [image](#), [lowPass5x\(\)](#), and [lowPass5y\(\)](#).

### 10.302.2.2 [TemplateMatchPyrBuilder](#) \* [TemplateMatchPyrBuilder::clone\(\)](#) `const` [virtual]

Cloning constructor useful to make clones from pointers:.

Implements [PyrBuilder< float >](#).

Definition at line 296 of file `PyrBuilder.C`.

The documentation for this class was generated from the following files:

- [PyrBuilder.H](#)
- [PyrBuilder.C](#)

## 10.303 TestSuite Class Reference

TestSuite manages a set of tests.

```
#include <TestSuite.H>
```

### Public Types

- enum [Op](#) {  
    **EQ, NEQ, LT, LTE,**  
    **GT, GTE** }  
    *Different types of arithmetic conditions that can be required.*

### Public Member Functions

- [TestSuite](#) ()  
    *Default constructor builds an empty test suite.*
- [~TestSuite](#) ()  
    *Destructor.*
- void [addTest](#) (const char \*name, TestFunc func)  
    *Add a test to the test suite.*
- void [printAvailableTests](#) () const  
    *Print all the available tests to stdout.*
- void [runTest](#) (int test\_n, int repeat\_n=1)  
    *Run a single test, optionally repeating it multiple times.*
- void [parseAndRun](#) (int argc, const char \*\*argv)  
    *Parse command-line args and take appropriate action.*
- void [require](#) (bool expr, int line, const char \*expr\_str)  
    *Require that the expression evaluates to true.*
- void [requireEq](#) (int expr, int expected, int line, const char \*expr\_str)  
    *Check that the given "int" expressions compare equal.*
- void [requireEq](#) (uint expr, uint expected, int line, const char \*expr\_str)  
    *Check that the given "uint" expressions compare equal.*

- void [requireEq](#) (long expr, long expected, int line, const char \*expr\_str)  
*Check that the given "long" expressions compare equal.*
- void [requireEq](#) (double expr, double expected, int line, const char \*expr\_str)  
*Check that the given "double" expressions compare equal.*
- void [require](#) (int lhs, [Op](#) op, int rhs, int line, const char \*lhs\_str, const char \*rhs\_str)  
*Check that the expression "lhs OP rhs" is true.*
- void [require](#) (long lhs, [Op](#) op, long rhs, int line, const char \*lhs\_str, const char \*rhs\_str)  
*Check that the expression "lhs OP rhs" is true.*
- void [require](#) (double lhs, [Op](#) op, double rhs, int line, const char \*lhs\_str, const char \*rhs\_str)  
*Check that the expression "lhs OP rhs" is true.*
- template<class T\_or\_RGB> void [requireEq](#) (const [Image](#)< T\_or\_RGB > &expr, const [Image](#)< T\_or\_RGB > &expected, int line, const char \*expr\_str)  
*Check that the given [Image](#) objects compare equal.*
- void [requireEq](#) (const [Image](#)< float > &expr, const [Image](#)< float > &expected, const float prec, int line, const char \*expr\_str)  
*Check that the given [Image](#) objects compare equal to given precision.*

### 10.303.1 Detailed Description

TestSuite manages a set of tests.

The tests can be run one at a time or all at once. Each "test" is a function whose signature matches the TestFunc typedef. Such functions receive a TestSuite as a reference parameter. This allows the TestFunc's to use TestSuite's diagnostic functions (such as [require\(\)](#) and [requireEq\(\)](#)) to verify whatever the test is testing.

Definition at line 53 of file TestSuite.H.

### 10.303.2 Member Enumeration Documentation

#### 10.303.2.1 enum [TestSuite::Op](#)

Different types of arithmetic conditions that can be required.

Definition at line 92 of file TestSuite.H.

### 10.303.3 Constructor & Destructor Documentation

#### 10.303.3.1 TestSuite::TestSuite ()

Default constructor builds an empty test suite.

Definition at line 176 of file TestSuite.C.

#### 10.303.3.2 TestSuite::~TestSuite ()

Destructor.

Definition at line 181 of file TestSuite.C.

### 10.303.4 Member Function Documentation

#### 10.303.4.1 void TestSuite::addTest (const char \* *name*, TestFunc *func*)

Add a test to the test suite.

(Note that the ADD\_TEST macro below makes it easier to have the string name of the function be the same as the C++ function name, without having to type it twice.)

Definition at line 187 of file TestSuite.C.

#### 10.303.4.2 void TestSuite::parseAndRun (int *argc*, const char \*\* *argv*)

Parse command-line args and take appropriate action.

This may involve either printing a list of available tests, or running a specific test.

Definition at line 212 of file TestSuite.C.

References printAvailableTests(), and runTest().

#### 10.303.4.3 void TestSuite::printAvailableTests () const

Print all the available tests to stdout.

Definition at line 193 of file TestSuite.C.

Referenced by parseAndRun().

**10.303.4.4** `void TestSuite::require (double lhs, Op op, double rhs, int line, const char * lhs_str, const char * rhs_str)`

Check that the expression "lhs OP rhs" is true.

Definition at line 286 of file TestSuite.C.

**10.303.4.5** `void TestSuite::require (long lhs, Op op, long rhs, int line, const char * lhs_str, const char * rhs_str)`

Check that the expression "lhs OP rhs" is true.

Definition at line 282 of file TestSuite.C.

**10.303.4.6** `void TestSuite::require (int lhs, Op op, int rhs, int line, const char * lhs_str, const char * rhs_str)`

Check that the expression "lhs OP rhs" is true.

Definition at line 278 of file TestSuite.C.

**10.303.4.7** `void TestSuite::require (bool expr, int line, const char * expr_str)`

Require that the expression evaluates to true.

Definition at line 240 of file TestSuite.C.

**10.303.4.8** `void TestSuite::requireEq (const Image< float > & expr, const Image< float > & expected, const float prec, int line, const char * expr_str)`

Check that the given **Image** objects compare equal to given precision.

Definition at line 272 of file TestSuite.C.

**10.303.4.9** `template<class T> void TestSuite::requireEq (const Image< T_or_RGB > & expr, const Image< T_or_RGB > & expected, int line, const char * expr_str)`

Check that the given **Image** objects compare equal.

Definition at line 268 of file TestSuite.C.

**10.303.4.10 void TestSuite::requireEq (double *expr*, double *expected*, int *line*, const char \* *expr\_str*)**

Check that the given "double" expressions compare equal.

Definition at line 263 of file TestSuite.C.

**10.303.4.11 void TestSuite::requireEq (long *expr*, long *expected*, int *line*, const char \* *expr\_str*)**

Check that the given "long" expressions compare equal.

Definition at line 259 of file TestSuite.C.

**10.303.4.12 void TestSuite::requireEq (uint *expr*, uint *expected*, int *line*, const char \* *expr\_str*)**

Check that the given "uint" expressions compare equal.

Definition at line 255 of file TestSuite.C.

References uint.

**10.303.4.13 void TestSuite::requireEq (int *expr*, int *expected*, int *line*, const char \* *expr\_str*)**

Check that the given "int" expressions compare equal.

Definition at line 251 of file TestSuite.C.

**10.303.4.14 void TestSuite::runTest (int *test\_n*, int *repeat\_n* = 1)**

Run a single test, optionally repeating it multiple times.

(Multiple repeats may be useful for profiling.) The output of the test is printed to standard output: on the first line is a boolean code indicating whether the test succeeded (code==1) or failed (code==0), and on subsequent lines are any diagnostic messages that may relate to the success or failure of the test (e.g., [require\(\)](#) and [requireEq\(\)](#) will generate a diagnostic message on failure).

Definition at line 200 of file TestSuite.C.

Referenced by [parseAndRun\(\)](#).

The documentation for this class was generated from the following files:

- [TestSuite.H](#)



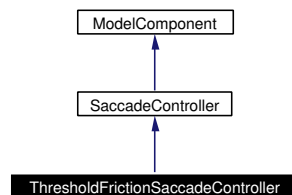
- [TestSuite.C](#)

## 10.304 ThresholdFrictionSaccadeController Class Reference

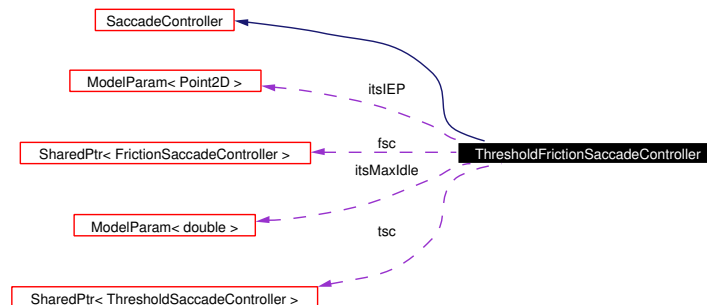
Threshold-limited friction-damped controller.

```
#include <SaccadeControllers.H>
```

Inheritance diagram for ThresholdFrictionSaccadeController:



Collaboration diagram for ThresholdFrictionSaccadeController:



### Public Member Functions

- [ThresholdFrictionSaccadeController](#) ([ModelManager](#) &mgr, const [std::string](#) paramprefix="SCeye")

*Constructor.*

- virtual [~ThresholdFrictionSaccadeController](#) ()

*Destructor.*

- virtual void [exportOptions](#) (const int whatToExport, const bool recurse=true)

*Export some of our [ModelParam](#) parameters as command-line-options.*

- virtual void [evolve](#) (const double t)  
*evolve one time step (called, e.g., by [Brain::evolve\(\)](#))*

## Protected Member Functions

- virtual void [computeWhenNewPercept](#) ()  
*This method is called each time a new percept has arrived.*
- virtual void [computeWhenResetPos](#) ()  
*This method is called each time a reset is made.*
- virtual [Point2D](#) [computeWhenNewDecision](#) (const double t)  
*This method is called each time a new decision is requested.*

## Protected Attributes

- [ModelParam](#)< double > [itsMaxIdle](#)  
*max time (in s) before we return to IEP*
- [ModelParam](#)< [Point2D](#) > [itsIEP](#)  
*homing position if idle*

### 10.304.1 Detailed Description

Threshold-limited friction-damped controller.

ThresholdFrictionSaccadeController combines the features of [ThresholdSaccadeController](#) and [FrictionSaccadeController](#).

The [ThresholdSaccadeController](#) determines when a new saccade is warranted. The [FrictionSaccadeController](#) is fed with the average covert location over the duration of the percept queue of the Threshold controller. It then is responsible for inducing slow drifts of overt attention towards that average, in between two saccades. In addition, this controller will return to the center of the image if it does not move in some time.

Definition at line 324 of file SaccadeControllers.H.

## 10.304.2 Constructor & Destructor Documentation

### 10.304.2.1 ThresholdFrictionSaccadeController::ThresholdFrictionSaccadeController ([ModelManager](#) & mgr, const [std::string](#) paramprefix = "SCeye")

Constructor.

#### Parameters:

*paramprefix* As this controller may be used either for the eye or for the head (as part of bigger controllers that include both eye and head), this should be either "SCeye" or "SChead" and will determine which model options we use for our internal parameters (eye or head)

Definition at line 518 of file SaccadeControllers.C.

References [ModelComponent::addSubComponent\(\)](#).

### 10.304.2.2 ThresholdFrictionSaccadeController::~~ThresholdFrictionSaccadeController () [virtual]

Destructor.

Definition at line 532 of file SaccadeControllers.C.

## 10.304.3 Member Function Documentation

### 10.304.3.1 [Point2D](#) ThresholdFrictionSaccadeController::computeWhenNewDecision (const double t) [protected, virtual]

This method is called each time a new decision is requested.

Implements [SaccadeController](#).

Definition at line 604 of file SaccadeControllers.C.

References [SaccadeController::beginSaccade\(\)](#), [SaccadeController::getPreviousDecision\(\)](#), [ModelParam< Point2D >::getVal\(\)](#), [ModelParam< double >::getVal\(\)](#), [SaccadeController::haveDecisions\(\)](#), [Point2D::i](#), [itsMaxIdle](#), [Point2DT::p](#), and [Point2DT::t](#).

### 10.304.3.2 void ThresholdFrictionSaccadeController::computeWhenNewPercept () [protected, virtual]

This method is called each time a new percept has arrived.

Implements [SaccadeController](#).

Definition at line 562 of file SaccadeControllers.C.

References [SaccadeController::getPreviousPercept\(\)](#), [Point2D::i](#), [Point2D::j](#), and [WTAWinner::p](#).

#### 10.304.3.3 void ThresholdFrictionSaccadeController::computeWhenResetPos() [protected, virtual]

This method is called each time a reset is made.

Implements [SaccadeController](#).

Definition at line 586 of file SaccadeControllers.C.

References [SaccadeController::beginSaccade\(\)](#), [SaccadeController::getPreviousPercept\(\)](#), [WTAWinner::p](#), and [WTAWinner::t](#).

#### 10.304.3.4 void ThresholdFrictionSaccadeController::evolve (const double t) [virtual]

evolve one time step (called, e.g., by [Brain::evolve\(\)](#))

Implements [SaccadeController](#).

Definition at line 549 of file SaccadeControllers.C.

References [SaccadeController::endSaccade\(\)](#), and [SaccadeController::isInSaccade\(\)](#).

#### 10.304.3.5 void ThresholdFrictionSaccadeController::exportOptions (const int whatToExport, const bool recurse = true) [virtual]

Export some of our [ModelParam](#) parameters as command-line-options.

See the base function in [ModelComponent.H](#) for info.

Reimplemented from [ModelComponent](#).

Definition at line 536 of file SaccadeControllers.C.

References [ModelComponent::exportOptions\(\)](#), [itsMaxIdle](#), [OPTEXP\\_CORE](#), and [ModelManager::requestOption\(\)](#).

### 10.304.4 Member Data Documentation

#### 10.304.4.1 [ModelParam<Point2D> ThresholdFrictionSaccadeController::its-IEP](#) [protected]

homing position if idle

Definition at line 362 of file SaccadeControllers.H.

#### 10.304.4.2 [ModelParam<double> ThresholdFrictionSaccadeController::its-MaxIdle](#) [protected]

max time (in s) before we return to IEP

Definition at line 361 of file SaccadeControllers.H.

Referenced by [computeWhenNewDecision\(\)](#), and [exportOptions\(\)](#).

The documentation for this class was generated from the following files:

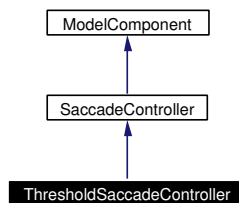
- [SaccadeControllers.H](#)
- [SaccadeControllers.C](#)

## 10.305 ThresholdSaccadeController Class Reference

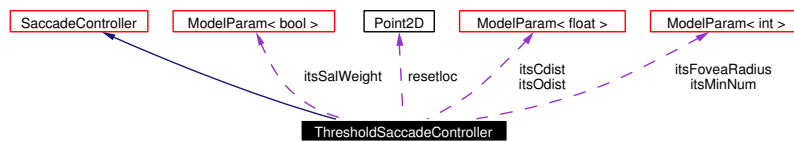
Threshold-limited controller that eliminates small or unreliable shifts.

```
#include <SaccadeControllers.H>
```

Inheritance diagram for ThresholdSaccadeController:



Collaboration diagram for ThresholdSaccadeController:



### Public Member Functions

- [ThresholdSaccadeController](#) ([ModelManager](#) &mgr, const [std::string](#) paramprefix="SCeye")  
*Constructor.*
- virtual [~ThresholdSaccadeController](#) ()  
*Destructor.*
- virtual void [exportOptions](#) (const int whatToExport, const bool recurse=true)  
*Export some of our [ModelParam](#) parameters as command-line-options.*
- virtual void [reset](#) (const bool recurse=true)  
*Reset ThresholdSaccadeController.*
- virtual void [evolve](#) (const double t)  
*evolve one time step (called, e.g., by [Brain::evolve\(\)](#))*

- virtual void [checkPercepts](#) (bool &areclose, [Point2D](#) &avgp)  
*Check queued-up percepts and return average.*

## Protected Member Functions

- virtual void [computeWhenNewPercept](#) ()  
*This method is called each time a new percept has arrived.*
- virtual void [computeWhenResetPos](#) ()  
*This method is called each time a reset is made.*
- virtual [Point2D](#) [computeWhenNewDecision](#) (const double t)  
*This method is called each time a new decision is requested.*
- virtual void [start1](#) ()  
*Get us started.*

## Protected Attributes

- [ModelParam](#)< float > [itsOdist](#)  
*min overt distance*
- [ModelParam](#)< float > [itsCdist](#)  
*max covert distance*
- [ModelParam](#)< int > [itsMinNum](#)  
*min number of close covert shifts*
- [ModelParam](#)< bool > [itsSalWeight](#)  
*weight average covert pos by saliency*
- [ModelParam](#)< int > [itsFoveaRadius](#)  
*radius of the fovea*
- float **odistsq**
- float **cdistsq**
- bool **sweight**
- bool **didresetpos**
- [Point2D](#) **resetloc**



### 10.305.1 Detailed Description

Threshold-limited controller that eliminates small or unreliable shifts.

ThresholdSaccadeController will issue an overt control whenever the received covert shifts are large and go to a same target for several successive covert shifts. It is hence to be used with IOR turned off, so that the covert system will keep attending to a same location for a while. Once a minimum number of covert shifts have been made that are sufficiently far from the current overt location and that are sufficiently close to each other, an overt shift will be issued, to the average location of the accumulated covert locations.

Definition at line 243 of file SaccadeControllers.H.

### 10.305.2 Constructor & Destructor Documentation

#### 10.305.2.1 ThresholdSaccadeController::ThresholdSaccadeController (ModelManager & mgr, const std::string paramprefix = "SCeye")

Constructor.

##### Parameters:

*paramprefix* As this controller may be used either for the eye or for the head (as part of bigger controllers that include both eye and head), this should be either "SCeye" or "SChead" and will determine which model options we use for our internal parameters (eye or head)

Definition at line 387 of file SaccadeControllers.C.

#### 10.305.2.2 ThresholdSaccadeController::~~ThresholdSaccadeController () [virtual]

Destructor.

Definition at line 410 of file SaccadeControllers.C.

### 10.305.3 Member Function Documentation

#### 10.305.3.1 void ThresholdSaccadeController::checkPercepts (bool & areclose, Point2D & avgp) [virtual]

Check queued-up percepts and return average.

check whether queued-up percepts are close to each other, and also return average position of percept queue

Definition at line 445 of file SaccadeControllers.C.

References `SaccadeController::getPreviousPercept()`, `ModelParam< bool >::getVal()`, `Point2D::i`, `WTAwinner::isValid()`, `itsSalWeight`, `Point2D::j`, `WTAwinner::p`, `Point2D::sqdist()`, and `WTAwinner::sv`.

Referenced by `computeWhenNewDecision()`.

### 10.305.3.2 **Point2D** `ThresholdSaccadeController::computeWhenNewDecision (const double t)` [protected, virtual]

This method is called each time a new decision is requested.

Implements [SaccadeController](#).

Definition at line 486 of file SaccadeControllers.C.

References `SaccadeController::beginSaccade()`, `checkPercepts()`, `SaccadeController::getPreviousDecision()`, `SaccadeController::haveDecisions()`, `SaccadeController::havePercepts()`, `Point2DT::p`, and `Point2D::sqdist()`.

### 10.305.3.3 **void** `ThresholdSaccadeController::computeWhenNewPercept ()` [protected, virtual]

This method is called each time a new percept has arrived.

Implements [SaccadeController](#).

Definition at line 471 of file SaccadeControllers.C.

### 10.305.3.4 **void** `ThresholdSaccadeController::computeWhenResetPos ()` [protected, virtual]

This method is called each time a reset is made.

Implements [SaccadeController](#).

Definition at line 475 of file SaccadeControllers.C.

References `SaccadeController::beginSaccade()`, `SaccadeController::getPreviousPercept()`, and `WTAwinner::p`.

### 10.305.3.5 **void** `ThresholdSaccadeController::evolve (const double t)` [virtual]

evolve one time step (called, e.g., by [Brain::evolve\(\)](#))

Implements [SaccadeController](#).

Definition at line 431 of file SaccadeControllers.C.

References `SaccadeController::endSaccade()`, and `SaccadeController::isInSaccade()`.

#### 10.305.3.6 `void ThresholdSaccadeController::exportOptions (const int whatToExport, const bool recurse = true) [virtual]`

Export some of our [ModelParam](#) parameters as command-line-options.

See the base function in [ModelComponent.H](#) for info.

Reimplemented from [ModelComponent](#).

Definition at line 414 of file SaccadeControllers.C.

References `ModelComponent::exportOptions()`, `itsCdist`, `itsMinNum`, `itsOdist`, `itsSalWeight`, `OPTEXP_CORE`, and `ModelManager::requestOption()`.

#### 10.305.3.7 `void ThresholdSaccadeController::reset (const bool recurse = true) [virtual]`

Reset `ThresholdSaccadeController`.

See the base function in [ModelComponent.H](#) for info.

Reimplemented from [SaccadeController](#).

Definition at line 435 of file SaccadeControllers.C.

References `SaccadeController::reset()`.

#### 10.305.3.8 `void ThresholdSaccadeController::start1 () [protected, virtual]`

Get us started.

Reimplemented from [ModelComponent](#).

Definition at line 399 of file SaccadeControllers.C.

References `ModelParam< float >::getVal()`, `ModelParam< int >::getVal()`, `itsCdist`, `itsMinNum`, `itsOdist`, and `SaccadeController::resetPqlen()`.

### 10.305.4 Member Data Documentation

#### 10.305.4.1 `ModelParam<float> ThresholdSaccadeController::itsCdist [protected]`

max covert distance

Definition at line 300 of file SaccadeControllers.H.

Referenced by exportOptions(), and start1().

**10.305.4.2   [ModelParam<int> ThresholdSaccadeController::itsFoveaRadius](#)  
[protected]**

radius of the fovea

Definition at line 303 of file SaccadeControllers.H.

**10.305.4.3   [ModelParam<int> ThresholdSaccadeController::itsMinNum](#)  
[protected]**

min number of close covert shifts

Definition at line 301 of file SaccadeControllers.H.

Referenced by exportOptions(), and start1().

**10.305.4.4   [ModelParam<float> ThresholdSaccadeController::itsOdists](#)  
[protected]**

min overt distance

Definition at line 299 of file SaccadeControllers.H.

Referenced by exportOptions(), and start1().

**10.305.4.5   [ModelParam<bool> ThresholdSaccadeController::itsSalWeight](#)  
[protected]**

weight average covert pos by saliency

Definition at line 302 of file SaccadeControllers.H.

Referenced by checkPercepts(), and exportOptions().

The documentation for this class was generated from the following files:

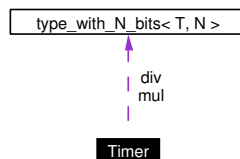
- [SaccadeControllers.H](#)
- [SaccadeControllers.C](#)

## 10.306 Timer Class Reference

This class implements a real-time timer.

```
#include <Timer.H>
```

Collaboration diagram for Timer:



### Public Member Functions

- `Timer` (const int tickspersec=1000)  
*Constructor; immediately start timing.*
- void `reset` ()  
*Reset the timer and immediately start timing again.*
- `uint64` `get` () const  
*Return elapsed time.*
- `uint64` `getReset` ()  
*Return elapsed time since last reset, and reset.*
- int `ticksPerSec` () const  
*Returns number of ticks per second, as specified at construction.*
- double `getSecs` ()  
*Returns number of seconds since last reset.*

### Protected Attributes

- `int64` `div`  
*time unit multipliers*
- `int64` `mul`

*time unit multipliers*

- timeval [tv](#)

*timestamp of last [reset\(\)](#)*

### 10.306.1 Detailed Description

This class implements a real-time timer.

Timing starts at construction or [reset\(\)](#).

At construction, you can decide on a number of timer ticks per second; use 1000 to count in milliseconds, 1000000 to count in microseconds, etc. The best resolution achievable is 1us, as this timer relies on the `gettimeofday()` system call, which has microsecond resolution. Use [get\(\)](#) to measure the number of elapsed ticks since last [reset\(\)](#), or [getSecs\(\)](#) to get the fractional number of seconds. Use [ticksPerSec\(\)](#) to retrieve this value, for example to convert yourself your measured times back to seconds instead of using [getSecs\(\)](#).

Definition at line 56 of file `Timer.H`.

### 10.306.2 Constructor & Destructor Documentation

#### 10.306.2.1 `Timer::Timer (const int tickpersec = 1000) [inline]`

Constructor; immediately start timing.

##### Parameters:

***tickpersec*** number of timer ticks per second; use 1000 to count in milliseconds, 1000000 to count in microseconds, etc. Values larger than 1000000 will throw a fatal error. If 1000000/tickpersec is not integer, a fatal error will also occur.

Definition at line 94 of file `Timer.H`.

References `int64`, and [reset\(\)](#).

### 10.306.3 Member Function Documentation

#### 10.306.3.1 `uint64 Timer::get () const [inline]`

Return elapsed time.

Return value is the number of ticks since last reset or since construction.

Definition at line 107 of file Timer.H.

References `div`, `int64`, `mul`, and `uint64`.

Referenced by `SingleChannelBeoServer::check()`, `extractBitObjects()`, `main()`, `BeobotVisualCortex::processEnd()`, `Staircase::pushEvent()`, `PsychoDisplay::pushEvent()`, `CINNIC::RunSimpleImage()`, `CarControl::toSpeedLinear()`, `CarControl::toSpeedSigmoid()`, and `PsychoDisplay::waitVsync()`.

#### 10.306.3.2 `uint64 Timer::getReset ()` [inline]

Return elapsed time since last reset, and reset.

Definition at line 115 of file Timer.H.

References `div`, `int64`, `mul`, and `uint64`.

Referenced by `main()`.

#### 10.306.3.3 `double Timer::getSecs ()` [inline]

Returns number of seconds since last reset.

Definition at line 129 of file Timer.H.

Referenced by `InferoTemporal::input()`, and `Beowulf::receive()`.

#### 10.306.3.4 `void Timer::reset ()` [inline]

Reset the timer and immediately start timing again.

Definition at line 103 of file Timer.H.

Referenced by `extractBitObjects()`, `main()`, `CarControl::rampSpeed()`, `Staircase::reset()`, `Beowulf::resetConnections()`, `CINNIC::RunSimpleImage()`, `Staircase::start2()`, `PsychoDisplay::start2()`, and `Timer()`.

#### 10.306.3.5 `int Timer::ticksPerSec () const` [inline]

Returns number of ticks per second, as specified at construction.

Definition at line 125 of file Timer.H.

References `mul`.

## 10.306.4 Member Data Documentation

### 10.306.4.1 [int64 Timer::div](#) [protected]

time unit multipliers

Definition at line 85 of file Timer.H.

Referenced by [get\(\)](#), and [getReset\(\)](#).

### 10.306.4.2 [int64 Timer::mul](#) [protected]

time unit multipliers

Definition at line 85 of file Timer.H.

Referenced by [get\(\)](#), [getReset\(\)](#), and [ticksPerSec\(\)](#).

### 10.306.4.3 [struct timeval Timer::tv](#) [protected]

timestamp of last [reset\(\)](#)

Definition at line 87 of file Timer.H.

The documentation for this class was generated from the following file:

- [Timer.H](#)

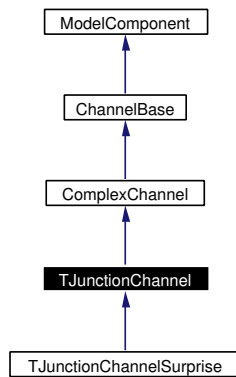


## 10.307 TJunctionChannel Class Reference

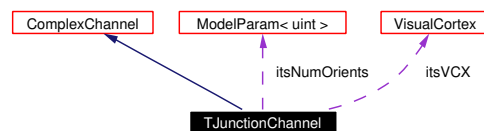
A T junction channel that detects T junctions at different.

```
#include <Channels.H>
```

Inheritance diagram for TJunctionChannel:



Collaboration diagram for TJunctionChannel:



### Public Member Functions

- [TJunctionChannel](#) ([ModelManager](#) &mgr, [VisualCortex](#) \*vcx)  
*Construct with standard params.*
- virtual [~TJunctionChannel](#) ()  
*Destructor.*
- virtual void [exportOptions](#) (const int whatToExport, const bool recurse=true)  
*Export some of our [ModelParam](#) parameters as command-line-options.*
- virtual bool [setModelParamString](#) (const [std::string](#) &name, const [std::string](#) &value, const bool recurse=false, const bool warn=true)

*Overload so that we can reconfigure when our params get changed.*

## Protected Member Functions

- virtual void `buildSubChans()`  
*(re-)build our subchannels*
- virtual void `doInput` (const `Image< PixRGB< float > > *`, const `Image< float > *bwimg`, const double `t`, const `Image< byte > &clipMask`)  
*Computes the T junctions at different orientations.*

## Protected Attributes

- `ModelParam< uint > itsNumOrients`  
*number of T junction subchannels*
- int `itsExportOpts`  
*keep track of what options to export in case we rebuild our subchans*
- `VisualCortex * itsVCX`

### 10.307.1 Detailed Description

A T junction channel that detects T junctions at different.

Definition at line 1323 of file Channels.H.

### 10.307.2 Constructor & Destructor Documentation

#### 10.307.2.1 `TJunctionChannel::TJunctionChannel (ModelManager & mgr, VisualCortex * vcx)`

Construct with standard params.

Definition at line 1898 of file Channels.C.

References `buildSubChans()`, `OPTEXP_NONE`, and `TJUNCTION`.

**10.307.2.2 TJunctionChannel::~~TJunctionChannel () [virtual]**

Destructor.

Definition at line 1909 of file Channels.C.

**10.307.3 Member Function Documentation****10.307.3.1 void TJunctionChannel::buildSubChans () [protected, virtual]**

(re-)build our subchannels

Reimplemented in [TJunctionChannelSurprise](#).

Definition at line 1913 of file Channels.C.

References [ComplexChannel::addSubChan\(\)](#), [ModelParam< uint >::getVal\(\)](#), [makeShared\(\)](#), [ModelComponent::removeAllSubComponents\(\)](#), [ChannelBase::setHomogeneous\(\)](#), [uint](#), and [ChannelBase::visualFeature\(\)](#).

Referenced by [setModelParamString\(\)](#), and [TJunctionChannel\(\)](#).

**10.307.3.2 void TJunctionChannel::doInput (const [Image](#)< [PixRGB](#)< float > > \*, const [Image](#)< float > \* *bwimg*, const double *t*, const [Image](#)< byte > & *clipMask*) [protected, virtual]**

Computes the T junctions at different orientations.

Implements [ChannelBase](#).

Definition at line 2005 of file Channels.C.

References [ASSERT](#), [VisualCortex::hasChannel\(\)](#), [ChannelBase::input\(\)](#), [ComplexChannel::numChans\(\)](#), [ComplexChannel::subChan\(\)](#), and [uint](#).

**10.307.3.3 void TJunctionChannel::exportOptions (const int *whatToExport*, const bool *recurse* = true) [virtual]**

Export some of our [ModelParam](#) parameters as command-line-options.

See the base function in [ModelComponent.H](#) for info.

Reimplemented from [ComplexChannel](#).

Definition at line 1990 of file Channels.C.

References [ComplexChannel::exportOptions\(\)](#), [OPTEXP\\_CORE](#), and [ModelManager::requestOption\(\)](#).

**10.307.3.4** `bool TJunctionChannel::setModelParamString (const std::string & name, const std::string & value, const bool recurse = false, const bool warn = true) [virtual]`

Overload so that we can reconfigure when our params get changed.

Reimplemented from [ModelComponent](#).

Definition at line 1972 of file Channels.C.

References [buildSubChans\(\)](#), [ModelParamBase::getName\(\)](#), [ModelParam< uint >::getVal\(\)](#), [ComplexChannel::numChans\(\)](#), and [ModelComponent::setModelParamString\(\)](#).

## 10.307.4 Member Data Documentation

**10.307.4.1** `int TJunctionChannel::itsExportOpts [protected]`

keep track of what options to export in case we rebuild our subchans

Definition at line 1355 of file Channels.H.

**10.307.4.2** `ModelParam<uint> TJunctionChannel::itsNumOrients [protected]`

number of T junction subchannels

Definition at line 1352 of file Channels.H.

The documentation for this class was generated from the following files:

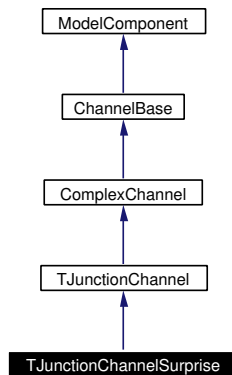
- [Channels.H](#)
- [Channels.C](#)

## 10.308 TJunctionChannelSurprise Class Reference

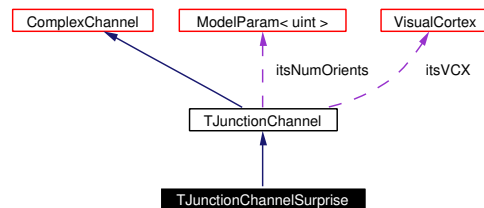
A T junction channel that detects T junctions at different orientations.

```
#include <ChannelsSurprise.H>
```

Inheritance diagram for TJunctionChannelSurprise:



Collaboration diagram for TJunctionChannelSurprise:



### Public Member Functions

- [TJunctionChannelSurprise](#) ([ModelManager](#) &mgr, [VisualCortex](#) \*vcx)

*Construct with standard params.*

- virtual [~TJunctionChannelSurprise](#) ()

*Destructor.*

## Protected Member Functions

- virtual void [buildSubChans](#) ()  
(re-)build our subchannels

### 10.308.1 Detailed Description

A T junction channel that detects T junctions at different orientations.  
Definition at line 580 of file ChannelsSurprise.H.

### 10.308.2 Constructor & Destructor Documentation

#### 10.308.2.1 [TJunctionChannelSurprise::TJunctionChannelSurprise](#) ([ModelManager](#) & mgr, [VisualCortex](#) \* vcx)

Construct with standard params.  
Definition at line 760 of file ChannelsSurprise.C.  
References [buildSubChans\(\)](#).

#### 10.308.2.2 [TJunctionChannelSurprise::~~TJunctionChannelSurprise](#) () [virtual]

Destructor.  
Definition at line 767 of file ChannelsSurprise.C.

### 10.308.3 Member Function Documentation

#### 10.308.3.1 [void TJunctionChannelSurprise::buildSubChans](#) () [protected, virtual]

(re-)build our subchannels  
Reimplemented from [TJunctionChannel](#).  
Definition at line 771 of file ChannelsSurprise.C.  
References [ComplexChannel::addSubChan\(\)](#), [ModelParam< uint >::getVal\(\)](#), [makeShared\(\)](#), [ModelComponent::removeAllSubComponents\(\)](#), [ChannelBase::setHomogeneous\(\)](#), [uint](#), and [ChannelBase::visualFeature\(\)](#).  
Referenced by [TJunctionChannelSurprise\(\)](#).

The documentation for this class was generated from the following files:

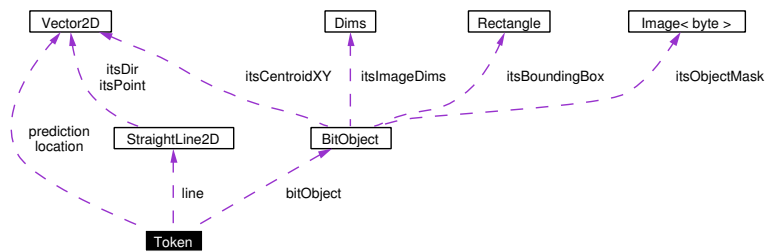
- [ChannelsSurprise.H](#)
- ChannelsSurprise.C

## 10.309 Token Class Reference

public class that contains information for a visual token for tracking

```
#include <VisualEvent.H>
```

Collaboration diagram for Token:



### Public Member Functions

- [Token](#) ()  
*default constructor*
- [Token](#) (float x, float y, [uint](#) frame, [BitObject](#) bo)  
*constructor that fills the members with initial values*
- [Token](#) ([BitObject](#) bo, [uint](#) frame)  
*constructor with the location being the centroid of the [BitObject](#)*
- [Token](#) (std::istream &is)  
*read the Token from the input stream is*
- void [writeToStream](#) (std::ostream &os) const  
*write the entire Token to the output stream os*
- void [readFromStream](#) (std::istream &is)  
*read the Token from the input stream is*
- void [writePosition](#) (std::ostream &os) const  
*write the Token's position to the output stream os*



## Public Attributes

- [BitObject bitObject](#)  
*the [BitObject](#) that holds info about the shape of this Token*
- [Vector2D location](#)  
*the x and y coordinates of the token in the frame*
- [Vector2D prediction](#)  
*the expected location for this token*
- [StraightLine2D line](#)  
*the straight line on which this token is moving*
- float [angle](#)  
*the angle between line and the connection to the FOE*
- [uint frame\\_nr](#)  
*the frame number at which the token occurred*

### 10.309.1 Detailed Description

public class that contains information for a visual token for tracking  
Definition at line 58 of file VisualEvent.H.

### 10.309.2 Constructor & Destructor Documentation

#### 10.309.2.1 Token::Token ()

default constructor  
Definition at line 53 of file VisualEvent.C.

#### 10.309.2.2 Token::Token (float x, float y, [uint frame](#), [BitObject bo](#))

constructor that fills the members with initial values  
Definition at line 63 of file VisualEvent.C.  
References [uint](#).

**10.309.2.3 Token::Token ([BitObject](#) bo, [uint](#) frame)**

constructor with the location being the centroid of the [BitObject](#)

Definition at line 73 of file VisualEvent.C.

References [uint](#).

**10.309.2.4 Token::Token (std::istream & is)**

read the Token from the input stream is

Definition at line 83 of file VisualEvent.C.

References [readFromStream\(\)](#).

**10.309.3 Member Function Documentation****10.309.3.1 void Token::readFromStream (std::istream & is)**

read the Token from the input stream is

Definition at line 101 of file VisualEvent.C.

References [angle](#), [bitObject](#), [frame\\_nr](#), [line](#), [location](#), [prediction](#), [StraightLine2D::readFromStream\(\)](#), and [Vector2D::readFromStream\(\)](#).

Referenced by [Token\(\)](#).

**10.309.3.2 void Token::writePosition (std::ostream & os) const**

write the Token's position to the output stream os

Definition at line 112 of file VisualEvent.C.

References [location](#), and [Vector2D::writeToStream\(\)](#).

**10.309.3.3 void Token::writeToStream (std::ostream & os) const**

write the entire Token to the output stream os

Definition at line 89 of file VisualEvent.C.

References [angle](#), [bitObject](#), [frame\\_nr](#), [line](#), [location](#), [prediction](#), [BitObject::writeToStream\(\)](#), [StraightLine2D::writeToStream\(\)](#), and [Vector2D::writeToStream\(\)](#).

## 10.309.4 Member Data Documentation

### 10.309.4.1 float [Token::angle](#)

the angle between line and the connection to the FOE

Definition at line 87 of file VisualEvent.H.

Referenced by `VisualEvent::getPropertyVector()`, `readFromStream()`, and `writeToStream()`.

### 10.309.4.2 [BitObject Token::bitObject](#)

the [BitObject](#) that holds info about the shape of this Token

Definition at line 75 of file VisualEvent.H.

Referenced by `VisualEvent::assign()`, `VisualEvent::doesIntersect()`, `VisualEventSet::drawTokens()`, `VisualEvent::getPropertyVector()`, `readFromStream()`, `MbariResultViewer::saveSingleEventFrame()`, `VisualEvent::VisualEvent()`, and `writeToStream()`.

### 10.309.4.3 [uint Token::frame\\_nr](#)

the frame number at which the token occurred

Definition at line 90 of file VisualEvent.H.

Referenced by `VisualEvent::assign()`, `VisualEvent::isTokenOk()`, `readFromStream()`, and `writeToStream()`.

### 10.309.4.4 [StraightLine2D Token::line](#)

the straight line on which this token is moving

Definition at line 84 of file VisualEvent.H.

Referenced by `readFromStream()`, and `writeToStream()`.

### 10.309.4.5 [Vector2D Token::location](#)

the x and y coordinates of the token in the frame

Definition at line 78 of file VisualEvent.H.

Referenced by `VisualEvent::assign()`, `VisualEventSet::drawTokens()`, `VisualEvent::getCost()`, `readFromStream()`, `VisualEvent::VisualEvent()`, `writePosition()`, and `writeToStream()`.

**10.309.4.6 [Vector2D Token::prediction](#)**

the expected location for this token

Definition at line 81 of file VisualEvent.H.

Referenced by `VisualEventSet::drawTokens()`, `readFromStream()`, and `writeToStream()`.

The documentation for this class was generated from the following files:

- [VisualEvent.H](#)
- [VisualEvent.C](#)

## 10.310 ToolkitInit Class Reference

A helper class for program-wide initialization.

```
#include <saliency.H>
```

### Public Member Functions

- [ToolkitInit](#) (bool fpe\_enabled)  
*Run global initialization routines if they have yet run.*
- [~ToolkitInit](#) ()  
*Run global shutdown routines if there are no more ToolkitInit objects around.*

### 10.310.1 Detailed Description

A helper class for program-wide initialization.

This makes a ToolkitInit object get instantiated in every translation unit (i.e., .C file), and thus the ToolkitInit's constructor is guaranteed to be run before any code in that .C file.

Definition at line 178 of file saliency.H.

### 10.310.2 Constructor & Destructor Documentation

#### 10.310.2.1 ToolkitInit::ToolkitInit (bool fpe\_enabled)

Run global initialization routines if they have yet run.

fpe\_enabled determines whether or not floating-point exceptions will be turned on.

Definition at line 79 of file saliency.C.

References `fpExceptionsOff()`, and `fpExceptionsOn()`.

#### 10.310.2.2 ToolkitInit::~~ToolkitInit ()

Run global shutdown routines if there are no more ToolkitInit objects around.

Definition at line 89 of file saliency.C.

References `ASSERT`.

The documentation for this class was generated from the following files:

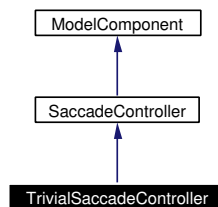
- [saliency.H](#)
- [saliency.C](#)

## 10.311 TrivialSaccadeController Class Reference

Trivial saccade controller.

```
#include <SaccadeControllers.H>
```

Inheritance diagram for TrivialSaccadeController:



Collaboration diagram for TrivialSaccadeController:

### Public Member Functions

- [TrivialSaccadeController](#) ([ModelManager](#) &mgr)  
*Constructor.*
- virtual [~TrivialSaccadeController](#) ()  
*Destructor.*
- virtual void [exportOptions](#) (const int whatToExport, const bool recurse=true)  
*Export some of our [ModelParam](#) parameters as command-line-options.*
- virtual void [evolve](#) (const double t)  
*evolve one time step (called, e.g., by [Brain::evolve\(\)](#))*

### Protected Member Functions

- virtual void [computeWhenNewPercept](#) ()  
*This method is called each time a new percept has arrived.*
- virtual void [computeWhenResetPos](#) ()  
*This method is called each time a reset is made.*
- virtual [Point2D](#) [computeWhenNewDecision](#) (const double t)

*This method is called each time a new decision is requested.*

### 10.311.1 Detailed Description

Trivial saccade controller.

TrivialSaccadeController is essentially a no-op (direct mapping from percept to decision).

Definition at line 103 of file SaccadeControllers.H.

### 10.311.2 Constructor & Destructor Documentation

#### 10.311.2.1 TrivialSaccadeController::TrivialSaccadeController ([ModelManager](#) & mgr)

Constructor.

Definition at line 147 of file SaccadeControllers.C.

#### 10.311.2.2 TrivialSaccadeController::~~TrivialSaccadeController () [virtual]

Destructor.

Definition at line 154 of file SaccadeControllers.C.

### 10.311.3 Member Function Documentation

#### 10.311.3.1 [Point2D](#) TrivialSaccadeController::computeWhenNewDecision (const double t) [protected, virtual]

This method is called each time a new decision is requested.

Implements [SaccadeController](#).

Definition at line 182 of file SaccadeControllers.C.

References [SaccadeController::getPreviousPercept\(\)](#), and [WTAwinner::p](#).

#### 10.311.3.2 void TrivialSaccadeController::computeWhenNewPercept () [protected, virtual]

This method is called each time a new percept has arrived.



Implements [SaccadeController](#).

Definition at line 174 of file SaccadeControllers.C.

#### 10.311.3.3 void TrivialSaccadeController::computeWhenResetPos () [protected, virtual]

This method is called each time a reset is made.

Implements [SaccadeController](#).

Definition at line 178 of file SaccadeControllers.C.

#### 10.311.3.4 void TrivialSaccadeController::evolve (const double t) [virtual]

evolve one time step (called, e.g., by [Brain::evolve\(\)](#))

Implements [SaccadeController](#).

Definition at line 170 of file SaccadeControllers.C.

#### 10.311.3.5 void TrivialSaccadeController::exportOptions (const int *whatToExport*, const bool *recurse* = true) [virtual]

Export some of our [ModelParam](#) parameters as command-line-options.

See the base function in [ModelComponent.H](#) for info.

Reimplemented from [ModelComponent](#).

Definition at line 158 of file SaccadeControllers.C.

References [ModelComponent::exportOptions\(\)](#).

The documentation for this class was generated from the following files:

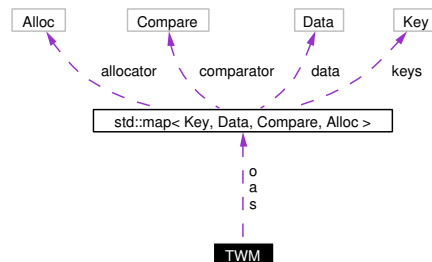
- [SaccadeControllers.H](#)
- [SaccadeControllers.C](#)

## 10.312 TWM Class Reference

Definition and access functions for TWM.

```
#include <Agent.H>
```

Collaboration diagram for TWM:



### Public Member Functions

- [TWM](#) ()  
*Constructor.*
- [~TWM](#) ()  
*Destructor.*
- bool [check\\_relevance](#) (char c, const [Entity](#) &entity, [std::vector](#)< [std::string](#) > &path)  
*check if entity is relevant to task graph*
- [Entity\\_TWM](#) & [add\\_compute\\_relevance](#) (char c, const [Entity](#) &entity)  
*add the new entity and find its relevance*
- void [propagate\\_relevance](#) (int path\_length, [Entity](#) &entity)  
*propagate relevance starting from given entity as root*
- void [insert](#) (char c, const [Entity](#) &entity)  
*insert entity to appropriate TWM*
- void [delete\\_element](#) (char c, [std::string](#) name)  
*delete entity from appropriate twm*
- [Entity\\_TWM](#) \* [get](#) (char c, [std::string](#) name)

*get the entity from appropriate twm*

### 10.312.1 Detailed Description

Definition and access functions for TWM.

The TWM class contains the object, subject and action task graphs. New entities are added to each graph and their relevance to the task are computed

Definition at line 185 of file Agent.H.

### 10.312.2 Constructor & Destructor Documentation

#### 10.312.2.1 TWM::TWM() [inline]

Constructor.

Definition at line 267 of file Agent.H.

#### 10.312.2.2 TWM::~~TWM() [inline]

Destructor.

Definition at line 271 of file Agent.H.

### 10.312.3 Member Function Documentation

#### 10.312.3.1 [Entity\\_TWM](#) & TWM::add\_compute\_relevance (char c, const [Entity](#) & entity)

add the new entity and find its relevance

**Parameters:**

*c* enter 'o' or 's' or 'a' depending on TWM

*entity* enter reference to the entity whose relevance needs to be checked

Definition at line 104 of file Agent.C.

References [get\(\)](#), [Entity::get\\_incoming\\_edges\(\)](#), [Entity::get\\_name\(\)](#), [Entity\\_TWM::get\\_relevance\(\)](#), [insert\(\)](#), and [Entity\\_TWM::set\\_relevance\(\)](#).

**10.312.3.2** `bool TWM::check_relevance (char c, const Entity & entity,  
std::vector< std::string > & path)`

check if entity is relevant to task graph

**Parameters:**

- c* enter 'o' or 's' or 'a' depending on TWM
- entity* enter the entity whose relevance needs to be checked
- path* enter a reference to the path from entity to task graph

Definition at line 563 of file Agent.C.

References `Entity::get_name()`.

**10.312.3.3** `void TWM::delete_element (char c, std::string name)`

delete entity from appropriate twm

**Parameters:**

- c* enter 'o' or 's' or 'a' depending on TWM
- name* enter the name of entity to be deleted

Definition at line 50 of file Agent.C.

**10.312.3.4** `Entity\_TWM * TWM::get (char c, std::string name)`

get the entity from appropriate twm

**Parameters:**

- c* enter 'o' or 's' or 'a' depending on TWM
- name* enter the name of entity to be deleted

Definition at line 76 of file Agent.C.

Referenced by `add_compute_relevance()`, and `insert()`.

**10.312.3.5** `void TWM::insert (char c, const Entity & entity)`

insert entity to appropriate TWM

**Parameters:**

- c* enter 'o' or 's' or 'a' depending on TWM

***entity*** enter a reference to entity to be inserted

Definition at line 692 of file Agent.C.

References `get()`, and `Entity::get_name()`.

Referenced by `add_compute_relevance()`.

#### 10.312.3.6 void TWM::propagate\_relevance (int *path\_length*, Entity & *entity*)

propagate relevance starting from given entity as root

**Parameters:**

***path\_length*** enter the depth upto which you want to propagate

***entity*** enter reference to the entity whose relevance needs to be checked

Definition at line 170 of file Agent.C.

References `Entity::get_name()`, `Entity::get_outgoing_edges_mutable()`, `Entity::get_relevance()`, and `Entity::set_relevance()`.

The documentation for this class was generated from the following files:

- Agent.H
- Agent.C

## 10.313 `type_if< T, B >` Struct Template Reference

Makes a *type* a typedef for *T* only if *B* is true.

```
#include <saliency.H>
```

### 10.313.1 Detailed Description

```
template<class T, bool B> struct type_if< T, B >
```

Makes a *type* a typedef for *T* only if *B* is true.

Definition at line 68 of file saliency.H.

The documentation for this struct was generated from the following file:

- [saliency.H](#)

## 10.314 `type_with_N_bits< T, N >` Struct Template Reference

A compile-time check that  $T$  has  $N$  bits.

```
#include <saliency.H>
```

### Public Types

- typedef [type\\_if](#)<  $T$ , `sizeof(T)*8==N` >::typ **type** )

#### 10.314.1 Detailed Description

```
template<class T, unsigned long N> struct type_with_N_bits< T, N >
```

A compile-time check that  $T$  has  $N$  bits.

Definition at line 75 of file `saliency.H`.

The documentation for this struct was generated from the following file:

- [saliency.H](#)

## 10.315 TypeTraits< T > Struct Template Reference

The generic TypeTraits implementation.

```
#include <TypeTraits.H>
```

### Public Types

- enum { **isTrivial** = 0 }

*Whether the type is trivial (either builtin or a plain-old-data struct).*

### 10.315.1 Detailed Description

**template<class T> struct TypeTraits< T >**

The generic TypeTraits implementation.

This template struct can be specialized or partially specialized for other user-defined types. In general, the approach should be that the generic version makes the most conservative specifications, and that specializations can make more aggressive specifications. That way, if we forget to specialize for a given type, we might risk missing some optimization, but won't risk breaking the program.

Definition at line 49 of file TypeTraits.H.

### 10.315.2 Member Enumeration Documentation

#### 10.315.2.1 template<class T> anonymous enum

Whether the type is trivial (either builtin or a plain-old-data struct).

This will be used to determine whether objects of the type can safely be handled as raw memory; that is, whether they can be initialized with `memset()`, copied with `memcpy()`, and don't have to have their destructors run, etc. Any of the builtin types meet these criteria, as well as any user-defined plain-old-data (POD) type. A POD type must have members that are all either builtin types or other POD types, and must not have a user-defined destructor, copy constructor, or assignment operator. It **can** have user-defined constructors.

Definition at line 61 of file TypeTraits.H.

The documentation for this struct was generated from the following file:

- [TypeTraits.H](#)



## 10.316 TypeTraits< PixRGB< T > > Struct Template Reference

Specialize [TypeTraits](#) to indicate that [PixRGB](#) is a "trivial" type.

```
#include <Pixels.H>
```

### Public Types

- enum { **isTrivial** = TypeTraits<T>::isTrivial }

#### 10.316.1 Detailed Description

```
template<class T> struct TypeTraits< PixRGB< T > >
```

Specialize [TypeTraits](#) to indicate that [PixRGB](#) is a "trivial" type.

That is, it is a type that can be safely initialized with `memset()`, and safely copied with `memcpy()`. In practical terms, this means we're relying on [PixRGB](#) not to do anything significant in its copy constructor or destructor.

Definition at line 53 of file `Pixels.H`.

The documentation for this struct was generated from the following file:

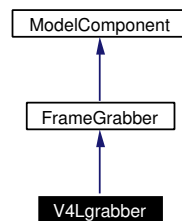
- [Pixels.H](#)

## 10.317 V4Lgrabber Class Reference

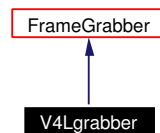
Definition and access functions for video4linux frame grabber.

```
#include <V4Lgrabber.H>
```

Inheritance diagram for V4Lgrabber:



Collaboration diagram for V4Lgrabber:



### Public Member Functions

- [V4Lgrabber](#) ([ModelManager](#) &mgr, const [std::string](#) &descrName="V4L Frame Grabber Driver", const [std::string](#) &tagName="V4LFrameGrabber")

*Constructor.*

- virtual [~V4Lgrabber](#) ()

*Destructor.*

- virtual [Image](#)< [PixRGB](#)< [byte](#) > > [grab](#) ()

*Grab a color image.*

- virtual [Image](#)< [PixRGB](#)< [byte](#) > > [grabSingle](#) ()

*Grab a single color image.*

## Protected Member Functions

- virtual void [start1](#) ()  
*get started*
- virtual void [stop2](#) ()  
*get stopped*

### 10.317.1 Detailed Description

Definition and access functions for video4linux frame grabber.

This class provides a trivial interface to Video4Linux frame grabbers. All the low-level setup is done during construction. The user only needs to call the [grab\(\)](#) method to capture an image. Shared memory and DMA directly from the grabber hardware into our memory is used if supported by the grabber and driver. This should work with any framegrabber that is supported by Video4Linux. We use Bt878-based boards.

After each grab, the next grab is initiated, and will be ready 33ms later (or one frame later if not NTSC). If you call [grab\(\)](#) again before 33ms have elapsed, it will block until the next frame is available. If you call it too late, you will have missed the latest frame, and [grab\(\)](#) will block until the next frame is acquired.

So a good strategy is to use [grab\(\)](#) to not only grab but also to synchronize your code with the video rate (30 frames/s if NTSC). Typically, then, you would have a main loop that first grabs and then does various processing that is guaranteed to take less than 33ms. You do not need to insert any pause after that processing to obtain a stable framerate; just finish your main loop, and the next call to [grab\(\)](#) (at the next iteration) will block until exactly one frame has passed since it was last called. See how this is done, for example, in `pvisionTCP-master.C` or `test-grab.C`

Definition at line 79 of file `V4Lgrabber.H`.

### 10.317.2 Constructor & Destructor Documentation

**10.317.2.1 V4Lgrabber::V4Lgrabber ([ModelManager](#) & *mgr*, const [std::string](#) & *descrName* = "V4L Frame Grabber Driver", const [std::string](#) & *tagName* = "V4LFrameGrabber")**

Constructor.

Definition at line 70 of file `V4Lgrabber.C`.

References `ModelParam< bool >::setVal()`, `ModelParam< FrameGrabberMode >::setVal()`, and `ModelParam< int >::setVal()`.

### 10.317.2.2 V4Lgrabber::~~V4Lgrabber () [virtual]

Destructor.

Definition at line 231 of file V4Lgrabber.C.

## 10.317.3 Member Function Documentation

### 10.317.3.1 Image< PixRGB< byte > > V4Lgrabber::grab () [virtual]

Grab a color image.

**Parameters:**

*image* Destination color image. Will be resized to grab width and height as specified in V4Lgrabber constructor.

Implements [FrameGrabber](#).

Definition at line 235 of file V4Lgrabber.C.

References [FrameGrabber::grab2rgb\(\)](#), and [image](#).

### 10.317.3.2 Image< PixRGB< byte > > V4Lgrabber::grabSingle () [virtual]

Grab a single color image.

This function will work better with single-shot grabs (as opposed to video) and USB cameras

Implements [FrameGrabber](#).

Definition at line 268 of file V4Lgrabber.C.

References [FrameGrabber::grab2rgb\(\)](#), and [image](#).

### 10.317.3.3 void V4Lgrabber::start1 () [protected, virtual]

get started

Reimplemented from [ModelComponent](#).

Definition at line 94 of file V4Lgrabber.C.

References [ModelParam< FrameGrabberMode >::getVal\(\)](#), [ModelParam< int >::getVal\(\)](#), [ModelParam< Dims >::getVal\(\)](#), [Dims::h\(\)](#), [V4LPalette::pal](#), [palettes](#), [ModelParam< FrameGrabberMode >::setVal\(\)](#), and [Dims::w\(\)](#).

**10.317.3.4 void V4Lgrabber::stop2 ()** [protected, virtual]

get stopped

Reimplemented from [ModelComponent](#).

Definition at line 224 of file V4Lgrabber.C.

The documentation for this class was generated from the following files:

- [V4Lgrabber.H](#)
- [V4Lgrabber.C](#)

## 10.318 V4LgrabberLight Class Reference

Wrapper class fro [V4Lgrabber](#) that restores the original interface.

```
#include <V4LgrabberLight.H>
```

Collaboration diagram for V4LgrabberLight:

### Public Member Functions

- [V4LgrabberLight](#) (const char \*device="/dev/video0", const int channel=1, const int subchan=0, const int ww=320, const int hh=240, const [FrameGrabberMode](#) gmode=GMRGB24, const bool byteswap=true, const float fps=30.0F, const int nbuf=2)

*Constructor.*

- [~V4LgrabberLight](#) ()

*Destructor.*

- [Image](#)< [PixRGB](#)< byte > > [grab](#) ()

*Grab a color image.*

### 10.318.1 Detailed Description

Wrapper class fro [V4Lgrabber](#) that restores the original interface.

This is convenient for simple programs that do not want to worry about all the Model-manager and [ModelComponent](#) stuff.

Definition at line 50 of file V4LgrabberLight.H.

### 10.318.2 Constructor & Destructor Documentation

**10.318.2.1** [V4LgrabberLight::V4LgrabberLight](#) (const char \* device = "/dev/video0", const int channel = 1, const int subchan = 0, const int ww = 320, const int hh = 240, const [FrameGrabberMode](#) gmode = GMRGB24, const bool byteswap = true, const float fps = 30.0F, const int nbuf = 2)

Constructor.

Definition at line 46 of file V4LgrabberLight.C.

References `ModelManager::exportOptions()`, `frameGrabberModeName()`, `OPTEXP_ALL`, `ModelManager::registerComponent()`, `ModelManager::setOptionValString()`, and `ModelComponent::start()`.

#### 10.318.2.2 V4LgrabberLight::~~V4LgrabberLight ()

Destructor.

Definition at line 84 of file `V4LgrabberLight.C`.

References `ModelComponent::stop()`.

### 10.318.3 Member Function Documentation

#### 10.318.3.1 `Image< PixRGB< byte > > V4LgrabberLight::grab ()`

Grab a color image.

**Parameters:**

*image* Destination color image. Will be resized to grab width and height as specified in `V4Lgrabber` constructor.

Definition at line 90 of file `V4LgrabberLight.C`.

The documentation for this class was generated from the following files:

- [V4LgrabberLight.H](#)
- [V4LgrabberLight.C](#)

## 10.319 V4LPalette Struct Reference

A V4L palette entry.

```
#include <V4Lgrabber.H>
```

### Public Attributes

- int [pal](#)  
*palette*
- double [bpp](#)
- int [depth](#)  
*depth*
- char \* [name](#)  
*name of palette e.g. "rgb24"*
- [FrameGrabberMode](#) [gm](#)  
*Frame grabber mode.*

### 10.319.1 Detailed Description

A V4L palette entry.

FIXME: reconcile this with [FrameGrabberMode.H](#)

Definition at line 123 of file V4Lgrabber.H.

### 10.319.2 Member Data Documentation

#### 10.319.2.1 double [V4LPalette::bpp](#)

bits per pixel, can be something like 1.5 in which case i/p bytes must be multiple of 2.

Definition at line 128 of file V4Lgrabber.H.

#### 10.319.2.2 int [V4LPalette::depth](#)

depth

Definition at line 129 of file V4Lgrabber.H.



**10.319.2.3** [FrameGrabberMode V4LPalette::gm](#)

Frame grabber mode.

Definition at line 131 of file V4Lgrabber.H.

**10.319.2.4** `char*` [V4LPalette::name](#)

name of palette e.g. "rgb24"

Definition at line 130 of file V4Lgrabber.H.

**10.319.2.5** `int` [V4LPalette::pal](#)

palette

Definition at line 125 of file V4Lgrabber.H.

Referenced by `V4Lgrabber::start1()`.

The documentation for this struct was generated from the following file:

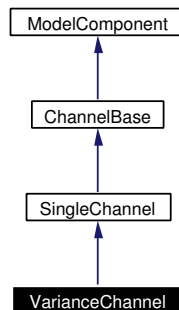
- [V4Lgrabber.H](#)

## 10.320 VarianceChannel Class Reference

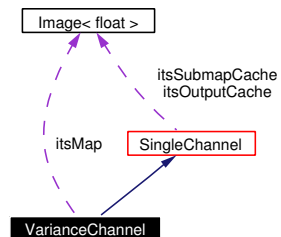
A variance channel, computes the variance in an image patch.

```
#include <Channels.H>
```

Inheritance diagram for VarianceChannel:



Collaboration diagram for VarianceChannel:



### Public Member Functions

- [VarianceChannel](#) ([ModelManager](#) &mgr)  
*Construct with standard params.*
- virtual [~VarianceChannel](#) ()  
*Destructor.*
- virtual bool [outputAvailable](#) () const  
*overload*

## Protected Member Functions

- virtual void [doInput](#) (const [Image](#)< [PixRGB](#)< float > > \*, const [Image](#)< float > \*bwimg, const double t, const [Image](#)< byte > &clipMask)

*special input function*

- virtual [Image](#)< float > [getOutput](#) ()

*special output function*

## Protected Attributes

- [Image](#)< float > [itsMap](#)

*our variance map*

### 10.320.1 Detailed Description

A variance channel, computes the variance in an image patch.

CAUTION: this channel will throw a fatal exception if your [LevelSpec](#) contains more than one level or if that level is different from the maplevel or if the deltas are non-zero.

Definition at line 506 of file Channels.H.

### 10.320.2 Constructor & Destructor Documentation

#### 10.320.2.1 VarianceChannel::VarianceChannel ([ModelManager](#) & mgr)

Construct with standard params.

Definition at line 2579 of file Channels.C.

#### 10.320.2.2 VarianceChannel::~VarianceChannel () [virtual]

Destructor.

Definition at line 2586 of file Channels.C.

### 10.320.3 Member Function Documentation

**10.320.3.1** `void VarianceChannel::doInput (const Image< PixRGB< float > > *, const Image< float > * bwimg, const double t, const Image< byte > & clipMask)` [protected, virtual]

special input function

Reimplemented from [SingleChannel](#).

Definition at line 2594 of file Channels.C.

References `ASSERT`, `Image< T >::begin()`, `Image< float >::beginw()`, `LevelSpec::delMax()`, `LevelSpec::delMin()`, `Image< T >::getHeight()`, `ModelParam< LevelSpec >::getVal()`, `Image< T >::getWidth()`, `LevelSpec::levMax()`, `LevelSpec::levMin()`, `LevelSpec::mapLevel()`, `Image< float >::resize()`, and `uint`.

**10.320.3.2** `Image< float > VarianceChannel::getOutput ()` [protected, virtual]

special output function

Reimplemented from [SingleChannel](#).

Definition at line 2646 of file Channels.C.

**10.320.3.3** `bool VarianceChannel::outputAvailable () const` [virtual]

overload

Reimplemented from [SingleChannel](#).

Definition at line 2590 of file Channels.C.

References `Image< float >::initialized()`.

### 10.320.4 Member Data Documentation

**10.320.4.1** `Image<float> VarianceChannel::itsMap` [protected]

our variance map

Definition at line 527 of file Channels.H.

The documentation for this class was generated from the following files:

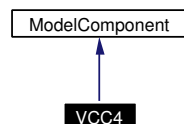
- [Channels.H](#)
- [Channels.C](#)

## 10.321 VCC4 Class Reference

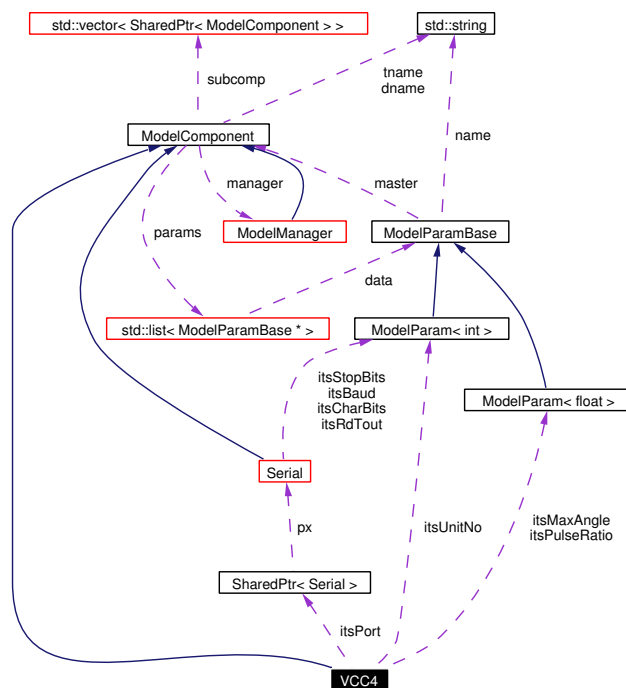
interface with the pan-tilt mechanism of a Canon VC-C4 camera

```
#include <VCC4.H>
```

Inheritance diagram for VCC4:



Collaboration diagram for VCC4:



### Public Member Functions

- **VCC4** (**ModelManager** &mgr, const **std::string** &descrName="VCC4 Pan/Tilt Camera Driver", const **std::string** &tagName="VCC4driver")

constructor. See [ModelComponent.H](#).

- [~VCC4](#) (void)  
*destructor - closes serial connection*
- void [exportOptions](#) (const int whatToExport, const bool recurse=true)  
*Export some of our [ModelParam](#) parameters as command-line-options.*
- int [gotoPosition](#) (float pan, float tilt, bool wait=true)  
*go to the specified position*
- int [getPosition](#) (float &pan, float &tilt, bool wait=true)  
*retrieve the current position*
- bool [IsMoving](#) (void)  
*is the pedestal moving?*
- int [PlainCommand](#) (int cmdno, bool wait=true)  
*send a command that does not need parameters*
- int [AbsNumCommand](#) (int cmdno, unsigned int value, bool wait=true)  
*send a command that needs an integer number as a parameter*
- int [AngleCommand](#) (int cmdno, float angle1, float angle2, bool wait=true)  
*send a command that needs two angles as parameters*
- int [StringCommand](#) (int cmdno, char \*string, bool wait=true)  
*send a command that needs a string as a parameter*
- int [getAbsNum](#) (unsigned int length)  
*retrieve an absolute number from the ReplyString*
- int [getOneAngle](#) (float &angle)  
*retrieve one angle from the ReplyString*
- int [getTwoAngles](#) (float &angle1, float &angle2)  
*retrieve two angle values from ReplyString*
- int [CameraInitialize](#) (bool goHome=false)  
*initialize camera*
- int [setUnitNo](#) (int unum)

*set the unit number*

- int [getUnitNo](#) (void)  
*get the unit number*
- float [getPulseRatio](#) (void)  
*returns the pulse ratio for the pan and tilt mechanism*

## Public Attributes

- char [ReplyString](#) [30]  
*reply string from last command (i.e. status reports and other replies)*

## Protected Member Functions

- int [SendRawCommand](#) (int cmdno, char \*param=NULL)  
*send a command with parameters to the device*
- int [WaitRawCommand](#) (int cmdno, char \*param=NULL)  
*send command and keep trying if busy*
- int [getIndex](#) (int cmdno)  
*return the index for a command number*
- int [getReply](#) (void)  
*attempt to retrieve a reply*
- int [intToHexString](#) (unsigned int value, char \*buf, unsigned int length)  
*transforms an unsigned int to a string of hex digits*
- int [hexStringToInt](#) (char \*buf, unsigned int length)  
*transforms a string of hex digits into an int*
- void [start1](#) ()  
*get started (see [ModelComponent.H](#)). The serial port gets configured.*
- void [stop1](#) ()  
*get stopped (see [ModelComponent.H](#)). We return the unit to home position*

## Protected Attributes

- [SharedPtr< Serial > itsPort](#)  
*our serial port*
- [ModelParam< std::string > itsSerialDevice](#)  
*our serial device*
- [ModelParam< int > itsUnitNo](#)  
*our unit number*
- [ModelParam< float > itsPulseRatio](#)  
*our pulse ratio*
- [ModelParam< float > itsMaxAngle](#)  
*our max angle*

### 10.321.1 Detailed Description

interface with the pan-tilt mechanism of a Canon VC-C4 camera

Definition at line 75 of file VCC4.H.

### 10.321.2 Constructor & Destructor Documentation

#### 10.321.2.1 VCC4::VCC4 ([ModelManager](#) & mgr, const [std::string](#) & descrName = "VCC4 Pan/Tilt Camera Driver", const [std::string](#) & tagName = "VCC4driver")

constructor. See [ModelComponent.H](#).

Definition at line 47 of file VCC4.C.

References [ModelComponent::addSubComponent\(\)](#).

#### 10.321.2.2 VCC4::~~VCC4 (void)

destructor - closes serial connection

Definition at line 77 of file VCC4.C.



### 10.321.3 Member Function Documentation

#### 10.321.3.1 **int VCC4::AbsNumCommand (int *cmdno*, unsigned int *value*, bool *wait* = true)**

send a command that needs an integer number as a parameter

**Parameters:**

*cmdno* for command numbers see [VCC4cmddef.H](#)

*value* the integer value to set

*wait* wait for successful completion if true

**Returns:**

0 if successful, some error code otherwise

Definition at line 209 of file VCC4.C.

References [getIndex\(\)](#), [intToHexString\(\)](#), [SendRawCommand\(\)](#), and [WaitRawCommand\(\)](#).

#### 10.321.3.2 **int VCC4::AngleCommand (int *cmdno*, float *angle1*, float *angle2*, bool *wait* = true)**

send a command that needs two angles as parameters

**Parameters:**

*cmdno* for command numbers see [VCC4cmddef.H](#)

*angle1*

*angle2* the angles (in degrees) to set

*wait* wait for successful completion if true

**Returns:**

0 if successful, some error code otherwise

Definition at line 253 of file VCC4.C.

References [getIndex\(\)](#), [ModelParam< float >::getVal\(\)](#), [intToHexString\(\)](#), [itsMaxAngle](#), [itsPulseRatio](#), [SendRawCommand\(\)](#), and [WaitRawCommand\(\)](#).

Referenced by [gotoPosition\(\)](#).

**10.321.3.3 int VCC4::CameraInitialize (bool *goHome* = false)**

initialize camera

after successful completion, ReplyString contains the name of the device.

**Parameters:**

*goHome* put camera into Home position if true

**Returns:**

0 if successful, some error code otherwise

Definition at line 94 of file VCC4.C.

References [getAbsNum\(\)](#), [getOneAngle\(\)](#), [ModelParam< float >::getVal\(\)](#), [itsMaxAngle](#), [itsPulseRatio](#), [PlainCommand\(\)](#), and [ModelParam< float >::setVal\(\)](#).

**10.321.3.4 void VCC4::exportOptions (const int *whatToExport*, const bool *recurse* = true) [virtual]**

Export some of our [ModelParam](#) parameters as command-line-options.

See the base function in [ModelComponent.H](#) for info.

Reimplemented from [ModelComponent](#).

Definition at line 62 of file VCC4.C.

References [ModelComponent::exportOptions\(\)](#), [itsMaxAngle](#), [itsPulseRatio](#), [itsSerialDevice](#), [itsUnitNo](#), [OPTEXP\\_CORE](#), and [ModelManager::requestOption\(\)](#).

**10.321.3.5 int VCC4::getAbsNum (unsigned int *length*) [inline]**

retrieve an absolute number from the ReplyString

**Parameters:**

*the* length of the string representing the number

**Returns:**

the integer value extracted from the ReplyString

Definition at line 232 of file VCC4.H.

References [hexStringToInt\(\)](#), and [ReplyString](#).

Referenced by [CameraInitialize\(\)](#).

**10.321.3.6 int VCC4::getIndex (int *cmdno*) [protected]**

return the index for a command number

Definition at line 430 of file VCC4.C.

Referenced by AbsNumCommand(), AngleCommand(), PlainCommand(), SendRawCommand(), and StringCommand().

**10.321.3.7 int VCC4::getOneAngle (float & *angle*)**

retrieve one angle from the ReplyString

**Parameters:**

*angle* reference to write the extracted angle to

**Returns:**

0 if successful, some error code otherwise

Definition at line 225 of file VCC4.C.

References ModelParam< float >::getVal(), hexStringToInt(), itsPulseRatio, and ReplyString.

Referenced by CameraInitialize().

**10.321.3.8 int VCC4::getPosition (float & *pan*, float & *tilt*, bool *wait* = true)**

retrieve the current position

**Parameters:**

*pan* panning angle (in degrees)

*tilt* tilting angle (in degrees)

*wait* wait for the command to successfully complete if true

**Returns:**

0 if successful, some error code otherwise

Definition at line 187 of file VCC4.C.

References getTwoAngles(), and PlainCommand().

Referenced by gotoPosition().

**10.321.3.9 float VCC4::getPulseRatio (void) [inline]**

returns the pulse ratio for the pan and tilt mechanism

Definition at line 238 of file VCC4.H.

References ModelParam< float >::getVal(), and itsPulseRatio.

**10.321.3.10 int VCC4::getReply (void) [protected]**

attempt to retrieve a reply

Definition at line 391 of file VCC4.C.

References ReplyString.

Referenced by SendRawCommand().

**10.321.3.11 int VCC4::getTwoAngles (float & *angle1*, float & *angle2*)**

retrieve two angle values from ReplyString

**Parameters:**

*angle1*

*angle2* references to write the extracted angles to (in degrees)

**Returns:**

0 if successful, some error code otherwise

Definition at line 237 of file VCC4.C.

References ModelParam< float >::getVal(), hexStringToInt(), itsPulseRatio, and ReplyString.

Referenced by getPosition(), and gotoPosition().

**10.321.3.12 int VCC4::getUnitNo (void) [inline]**

get the unit number

Definition at line 219 of file VCC4.H.

References ModelParam< int >::getVal(), and itsUnitNo.

**10.321.3.13 int VCC4::gotoPosition (float *pan*, float *tilt*, bool *wait* = true)**

go to the specified position

**Parameters:**

- pan* panning angle (in degrees)
- tilt* tilting angle (in degrees)
- wait* wait for the command to successfully complete if true

**Returns:**

- 0 if successful, some error code otherwise

Definition at line 143 of file VCC4.C.

References AngleCommand(), getPosition(), getTwoAngles(), and PlainCommand().

**10.321.3.14 int VCC4::hexStringToInt (char \* *buf*, unsigned int *length*)**  
[protected]

transforms a string of hex digits into an int

Definition at line 314 of file VCC4.C.

Referenced by getAbsNum(), getOneAngle(), and getTwoAngles().

**10.321.3.15 int VCC4::intToHexString (unsigned int *value*, char \* *buf*, unsigned int *length*)** [protected]

transforms an unsigned int to a string of hex digits

Definition at line 295 of file VCC4.C.

Referenced by AbsNumCommand(), and AngleCommand().

**10.321.3.16 bool VCC4::IsMoving (void)** [inline]

is the pedestal moving?

Definition at line 225 of file VCC4.H.

References PlainCommand().

**10.321.3.17 int VCC4::PlainCommand (int *cmdno*, bool *wait* = true)**

send a command that does not need parameters

**Parameters:**

- cmdno* for command numbers see [VCC4cmddef.H](#)
- wait* wait for successful completion if true

**Returns:**

0 if successful, some error code otherwise

Definition at line 197 of file VCC4.C.

References [getIndex\(\)](#), [SendRawCommand\(\)](#), and [WaitRawCommand\(\)](#).

Referenced by [CameraInitialize\(\)](#), [getPosition\(\)](#), [gotoPosition\(\)](#), [IsMoving\(\)](#), and [stop1\(\)](#).

**10.321.3.18 int VCC4::SendRawCommand (int *cmdno*, char \* *param* = NULL) [protected]**

send a command with parameters to the device

Definition at line 346 of file VCC4.C.

References [getIndex\(\)](#), [getReply\(\)](#), [ModelParam< int >::getVal\(\)](#), and [itsUnitNo](#).

Referenced by [AbsNumCommand\(\)](#), [AngleCommand\(\)](#), [PlainCommand\(\)](#), [StringCommand\(\)](#), and [WaitRawCommand\(\)](#).

**10.321.3.19 int VCC4::setUnitNo (int *unum*) [inline]**

set the unit number

Definition at line 211 of file VCC4.H.

References [itsUnitNo](#), and [ModelParam< int >::setVal\(\)](#).

**10.321.3.20 void VCC4::start1 () [protected, virtual]**

get started (see [ModelComponent.H](#)). The serial port gets configured.

Reimplemented from [ModelComponent](#).

Definition at line 81 of file VCC4.C.

References [itsSerialDevice](#).

**10.321.3.21 void VCC4::stop1 () [protected, virtual]**

get stopped (see [ModelComponent.H](#)). We return the unit to home position

Reimplemented from [ModelComponent](#).

Definition at line 88 of file VCC4.C.

References [PlainCommand\(\)](#).

### 10.321.3.22 `int VCC4::StringCommand (int cmdno, char * string, bool wait = true)`

send a command that needs a string as a parameter

#### Parameters:

*cmdno* for command numbers see [VCC4cmddef.H](#)

*string* the string value to set

*wait* wait for successful completion if true

#### Returns:

0 if successful, some error code otherwise

Definition at line 277 of file VCC4.C.

References [getIndex\(\)](#), [SendRawCommand\(\)](#), and [WaitRawCommand\(\)](#).

### 10.321.3.23 `int VCC4::WaitRawCommand (int cmdno, char * param = NULL) [protected]`

send command and keep trying if busy

Definition at line 330 of file VCC4.C.

References [SendRawCommand\(\)](#).

Referenced by [AbsNumCommand\(\)](#), [AngleCommand\(\)](#), [PlainCommand\(\)](#), and [StringCommand\(\)](#).

## 10.321.4 Member Data Documentation

### 10.321.4.1 `ModelParam<float> VCC4::itsMaxAngle` [protected]

our max angle

Definition at line 175 of file VCC4.H.

Referenced by [AngleCommand\(\)](#), [CameraInitialize\(\)](#), and [exportOptions\(\)](#).

### 10.321.4.2 `SharedPtr<Serial> VCC4::itsPort` [protected]

our serial port

Definition at line 171 of file VCC4.H.

**10.321.4.3   [ModelParam<float>](#)   [VCC4::itsPulseRatio](#)   [protected]**

our pulse ratio

Definition at line 174 of file VCC4.H.

Referenced by [AngleCommand\(\)](#), [CameraInitialize\(\)](#), [exportOptions\(\)](#), [getOneAngle\(\)](#), [getPulseRatio\(\)](#), and [getTwoAngles\(\)](#).

**10.321.4.4   [ModelParam<std::string>](#)   [VCC4::itsSerialDevice](#)   [protected]**

our serial device

Definition at line 172 of file VCC4.H.

Referenced by [exportOptions\(\)](#), and [start1\(\)](#).

**10.321.4.5   [ModelParam<int>](#)   [VCC4::itsUnitNo](#)   [protected]**

our unit number

Definition at line 173 of file VCC4.H.

Referenced by [exportOptions\(\)](#), [getUnitNo\(\)](#), [SendRawCommand\(\)](#), and [setUnitNo\(\)](#).

**10.321.4.6   [char](#)   [VCC4::ReplyString](#)[30]**

reply string from last command (i.e. status reports and other replies)

Definition at line 168 of file VCC4.H.

Referenced by [getAbsNum\(\)](#), [getOneAngle\(\)](#), [getReply\(\)](#), and [getTwoAngles\(\)](#).

The documentation for this class was generated from the following files:

- [VCC4.H](#)
- [VCC4.C](#)

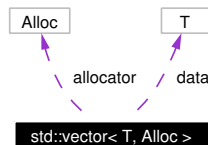


## 10.322 `std::vector< T, Alloc >` Class Template Reference

The STL vector container class. Use with `#include` .

```
#include <dummySTL.H>
```

Collaboration diagram for `std::vector< T, Alloc >`:



### Public Attributes

- `T data`
- `Alloc allocator`

#### 10.322.1 Detailed Description

```
template<class T, class Alloc> class std::vector< T, Alloc >
```

The STL vector container class. Use with `#include` .

See [SGI's documentation](#) of `std::vector`.

Definition at line 13 of file `dummySTL.H`.

The documentation for this class was generated from the following file:

- [dummySTL.H](#)

## 10.323 Vector2D Class Reference

A vector in the 2D plane.

```
#include <Geometry2D.H>
```

### Public Member Functions

- [Vector2D](#) ()  
*default constructor, constructs an invalid Vector2D*
- [Vector2D](#) (float x, float y)  
*construct a Vector2D with coordinates x and y*
- [Vector2D](#) (const [Point2D](#) &point)  
*construct a Vector2D from a [Point2D](#)*
- [Vector2D](#) (std::istream &is)  
*Construct from the data contained in the input stream is.*
- void [reset](#) (float x, float y)  
*reset the entries of the vector*
- void [reset](#) (const [Point2D](#) &point)  
*reset the entries of the vector from a [Point2D](#)*
- void [writeToStream](#) (std::ostream &os) const  
*write the entire Vector2D to the output stream os*
- void [readFromStream](#) (std::istream &is)  
*read the Vector2D from the input stream is*
- float [x](#) () const  
*access the x coordinate*
- float [y](#) () const  
*access the y coordinate*
- [Point2D](#) [getPoint2D](#) () const  
*return a [Point2D](#) (rounding the coordinates to int)*
- float [dotProduct](#) (const [Vector2D](#) &other) const

*compute the dot product of this with other*

- float **length** () const  
*compute the length of the vector*
- float **normalize** ()  
*normalize the vector to length 1 and return its old length*
- float **distance** (const **Vector2D** &other) const  
*compute the Euclidean distance of this from other*
- float **angle** (const **Vector2D** &other) const  
*compute the angle (in degrees) between this and other*
- bool **isCollinear** (const **Vector2D** &other) const  
*determine whether this and other are collinear*
- bool **isOrthogonal** (const **Vector2D** &other) const  
*determine whether this and other are orthogonal*
- bool **isZero** () const  
*whether this vector is zero*
- bool **isValid** () const  
*whether this vector is valid*
- float **operator \*** (const **Vector2D** &v) const  
*dot product*
- **Vector2D operator+** (const **Vector2D** &v) const  
*component-based addition*
- **Vector2D operator-** (const **Vector2D** &v) const  
*component-based subtraction*
- **Vector2D & operator+=** (const **Vector2D** &v)  
*component-based addition*
- **Vector2D & operator-=** (const **Vector2D** &v)
- **Vector2D operator+** (const float f) const  
*scalar addition*
- **Vector2D operator-** (const float f) const

*scalar subtraction*

- [Vector2D operator \\*](#) (const float f) const

*scalar multiplication*

- [Vector2D operator/](#) (const float f) const

*scalar division*

- [Vector2D & operator+=](#) (const float f)

*scalar addition*

- [Vector2D & operator-=](#) (const float f)

*scalar subtraction*

- [Vector2D & operator \\*=](#) (const float f)

*scalar multiplication*

- [Vector2D & operator/=](#) (const float f)

*scalar division*

### 10.323.1 Detailed Description

A vector in the 2D plane.

Definition at line 48 of file Geometry2D.H.

### 10.323.2 Constructor & Destructor Documentation

#### 10.323.2.1 [Vector2D::Vector2D \(\)](#)

default constructor, constructs an invalid Vector2D

Definition at line 45 of file Geometry2D.C.

Referenced by [operator \\*\(\)](#), [operator +\(\)](#), [operator -\(\)](#), and [operator /\(\)](#).

#### 10.323.2.2 [Vector2D::Vector2D \(float x, float y\)](#)

construct a Vector2D with coordinates x and y

Definition at line 50 of file Geometry2D.C.

### 10.323.2.3 Vector2D::Vector2D (const [Point2D](#) & *point*)

construct a Vector2D from a [Point2D](#)

Definition at line 55 of file Geometry2D.C.

### 10.323.2.4 Vector2D::Vector2D (std::istream & *is*)

Construct from the data contained in the input stream *is*.

Definition at line 60 of file Geometry2D.C.

References `readFromStream()`.

## 10.323.3 Member Function Documentation

### 10.323.3.1 float Vector2D::angle (const [Vector2D](#) & *other*) const

compute the angle (in degrees) between this and other

Definition at line 155 of file Geometry2D.C.

References `ASSERT`, `dotProduct()`, `isValid()`, and `length()`.

Referenced by `isCollinear()`.

### 10.323.3.2 float Vector2D::distance (const [Vector2D](#) & *other*) const

compute the Euclidean distance of this from other

Definition at line 146 of file Geometry2D.C.

References `ASSERT`, `isValid()`, and `length()`.

### 10.323.3.3 float Vector2D::dotProduct (const [Vector2D](#) & *other*) const

compute the dot product of this with other

Definition at line 121 of file Geometry2D.C.

References `ASSERT`, `isValid()`, `x()`, and `y()`.

Referenced by `angle()`, `isOrthogonal()`, `length()`, and operator `*`.

### 10.323.3.4 [Point2D](#) Vector2D::getPoint2D () const

return a [Point2D](#) (rounding the coordinates to int)

Definition at line 114 of file Geometry2D.C.

References ASSERT, and isValid().

Referenced by VisualEventSet::drawTokens(), and BitObject::getCentroid().

#### **10.323.3.5 bool Vector2D::isCollinear (const [Vector2D](#) & other) const**

determine whether this and other are collinear

Definition at line 163 of file Geometry2D.C.

References angle(), ASSERT, and isValid().

Referenced by StraightLine2D::isParallel().

#### **10.323.3.6 bool Vector2D::isOrthogonal (const [Vector2D](#) & other) const**

determine whether this and other are orthogonal

Definition at line 171 of file Geometry2D.C.

References ASSERT, dotProduct(), and isValid().

Referenced by StraightLine2D::isOrthogonal().

#### **10.323.3.7 bool Vector2D::isValid () const**

whether this vector is valid

Definition at line 186 of file Geometry2D.C.

Referenced by angle(), VisualEvent::assign(), distance(), dotProduct(), VisualEventSet::drawTokens(), getPoint2D(), isCollinear(), isOrthogonal(), StraightLine2D::isPointOnLine(), isZero(), normalize(), operator\*(), operator+(), operator+=(), operator-(), operator/(), x(), and y().

#### **10.323.3.8 bool Vector2D::isZero () const**

whether this vector is zero

Definition at line 179 of file Geometry2D.C.

References ASSERT, and isValid().

#### **10.323.3.9 float Vector2D::length () const**

compute the length of the vector

Definition at line 129 of file Geometry2D.C.

References dotProduct(), and sqrt().

Referenced by angle(), distance(), and normalize().

#### 10.323.3.10 float Vector2D::normalize ()

normalize the vector to length 1 and return its old length

Definition at line 133 of file Geometry2D.C.

References ASSERT, isValid(), and length().

#### 10.323.3.11 Vector2D Vector2D::operator \* (const float f) const

scalar multiplication

Definition at line 240 of file Geometry2D.C.

References isValid(), and Vector2D().

#### 10.323.3.12 float Vector2D::operator \* (const Vector2D & v) const

dot product

Definition at line 192 of file Geometry2D.C.

References dotProduct().

#### 10.323.3.13 Vector2D & Vector2D::operator \*= (const float f)

scalar multiplication

Definition at line 270 of file Geometry2D.C.

#### 10.323.3.14 Vector2D Vector2D::operator+ (const float f) const

scalar addition

Definition at line 226 of file Geometry2D.C.

References isValid(), and Vector2D().

#### 10.323.3.15 Vector2D Vector2D::operator+ (const Vector2D & v) const

component-based addition

Definition at line 196 of file Geometry2D.C.

References isValid(), Vector2D(), x(), and y().

#### **10.323.3.16   [Vector2D](#) & Vector2D::operator+= (const float f)**

scalar addition

Definition at line 256 of file Geometry2D.C.

#### **10.323.3.17   [Vector2D](#) & Vector2D::operator+= (const [Vector2D](#) & v)**

component-based addition

Definition at line 210 of file Geometry2D.C.

References isValid(), x(), and y().

#### **10.323.3.18   [Vector2D](#) Vector2D::operator- (const float f) const**

scalar subtraction

Definition at line 233 of file Geometry2D.C.

References isValid(), and Vector2D().

#### **10.323.3.19   [Vector2D](#) Vector2D::operator- (const [Vector2D](#) & v) const**

component-based subtraction

Definition at line 203 of file Geometry2D.C.

References isValid(), Vector2D(), x(), and y().

#### **10.323.3.20   [Vector2D](#) & Vector2D::operator-= (const float f)**

scalar subtraction

Definition at line 263 of file Geometry2D.C.

#### **10.323.3.21   [Vector2D](#) Vector2D::operator/ (const float f) const**

scalar division

Definition at line 247 of file Geometry2D.C.

References ASSERT, isValid(), and Vector2D().



**10.323.3.22** [Vector2D](#) & [Vector2D::operator/=](#) (const float f)

scalar division

Definition at line 277 of file Geometry2D.C.

References ASSERT.

**10.323.3.23** void [Vector2D::readFromStream](#) (std::istream & is)

read the Vector2D from the input stream is

Definition at line 91 of file Geometry2D.C.

Referenced by [Token::readFromStream\(\)](#), [StraightLine2D::readFromStream\(\)](#), and [Vector2D\(\)](#).

**10.323.3.24** void [Vector2D::reset](#) (const [Point2D](#) & point)

reset the entries of the vector from a [Point2D](#)

Definition at line 75 of file Geometry2D.C.

References [Point2D::i](#), and [Point2D::j](#).

**10.323.3.25** void [Vector2D::reset](#) (float x, float y)

reset the entries of the vector

Definition at line 67 of file Geometry2D.C.

Referenced by [BitObject::reset\(\)](#), and [FOEestimator::updateFOE\(\)](#).

**10.323.3.26** void [Vector2D::writeToStream](#) (std::ostream & os) const

write the entire Vector2D to the output stream os

Definition at line 83 of file Geometry2D.C.

Referenced by [Token::writePosition\(\)](#), [Token::writeToStream\(\)](#), [StraightLine2D::writeToStream\(\)](#), and [BitObject::writeToStream\(\)](#).

**10.323.3.27** float [Vector2D::x](#) () const

access the x coordinate

Definition at line 100 of file Geometry2D.C.

References ASSERT, and isValid().

Referenced by VisualEvent::assign(), dotProduct(), VisualEvent::getCost(), StraightLine2D::intersect(), StraightLine2D::isPointOnLine(), operator!(), operator+(), operator+=(), operator-(), operator==(), and VisualEvent::VisualEvent().

#### 10.323.3.28 float Vector2D::y () const

access the y coordinate

Definition at line 107 of file Geometry2D.C.

References ASSERT, and isValid().

Referenced by VisualEvent::assign(), dotProduct(), VisualEvent::getCost(), StraightLine2D::intersect(), StraightLine2D::isPointOnLine(), operator!(), operator+(), operator+=(), operator-(), operator==(), and VisualEvent::VisualEvent().

The documentation for this class was generated from the following files:

- [Geometry2D.H](#)
- [Geometry2D.C](#)

## 10.324 Vector3 Class Reference

```
#include <Vector3.H>
```

### Public Member Functions

- [Vector3](#) ()
- [Vector3](#) (double a, double b, double c)
- [Vector3](#) (const [Vector3](#) &v)
- [Vector3](#) & [operator=](#) (const [Vector3](#) &v)
- void [Set](#) (double a, double b, double c)
- bool [operator==](#) (const [Vector3](#) &v) const
- bool [operator!=](#) (const [Vector3](#) &v) const
- bool [IsZero](#) () const
- bool [IsNearlyZero](#) () const
- bool [IsNearlyEqual](#) (const [Vector3](#) &v) const
- [Vector3](#) [operator+](#) (const [Vector3](#) &v) const
- [Vector3](#) [operator-](#) (const [Vector3](#) &v) const
- [Vector3](#) [operator\\*](#) (double Scale) const
- [Vector3](#) [operator/](#) (double Scale) const
- [Vector3](#) [operator-](#) () const
- [Vector3](#) [operator+=](#) (const [Vector3](#) &v)
- [Vector3](#) [operator-=](#) (const [Vector3](#) &v)
- [Vector3](#) [operator\\*+=](#) (double Scale)
- [Vector3](#) [operator/=](#) (double Scale)
- double [Norm](#) () const
- double [NormSquared](#) () const
- [Vector3](#) [Cross](#) (const [Vector3](#) &v) const
- double [Dot](#) (const [Vector3](#) &v) const
- bool [Normalize](#) ()
- [Vector3](#) [GetNormalized](#) () const

### Public Attributes

- double [x](#)
- double [y](#)
- double [z](#)

### Friends

- [Vector3](#) [operator\\*](#) (double Scale, const [Vector3](#) &v)

### 10.324.1 Detailed Description

general : Vector3 is a class for handling 3D Vectors manipulation.

We made a choose to store only a 3 dimensions vectors to speed up system when we change of view point. A general 3D transformation could be placed in a 4x4 matrix, but some of the coefficients are always null. So that's waste of time.

Definition at line 38 of file Vector3.H.

### 10.324.2 Constructor & Destructor Documentation

#### 10.324.2.1 Vector3::Vector3 () [inline]

default constructor

Definition at line 47 of file Vector3.H.

Referenced by Cross(), operator\*(), operator+(), operator-(), and operator/().

#### 10.324.2.2 Vector3::Vector3 (double *a*, double *b*, double *c*) [inline]

constructor with initialisation

Definition at line 50 of file Vector3.H.

#### 10.324.2.3 Vector3::Vector3 (const Vector3 & *v*) [inline]

copy constructor

Definition at line 53 of file Vector3.H.

References x, y, and z.

### 10.324.3 Member Function Documentation

#### 10.324.3.1 Vector3 Vector3::Cross (const Vector3 & *v*) const [inline]

cross product

Definition at line 184 of file Vector3.H.

References Vector3(), x, y, and z.

#### 10.324.3.2 double Vector3::Dot (const Vector3 & *v*) const [inline]

dot product

Definition at line 190 of file Vector3.H.

References x, y, and z.

#### 10.324.3.3 **Vector3** Vector3::GetNormalized () const [inline]

return a normalized vector

Definition at line 211 of file Vector3.H.

References sqrt(), x, y, and z.

#### 10.324.3.4 **bool** Vector3::IsNearlyEqual (const **Vector3** & v) const [inline]

comparison : nearly equal

Definition at line 92 of file Vector3.H.

References x, y, and z.

#### 10.324.3.5 **bool** Vector3::IsNearlyZero () const [inline]

comparison : nearly zero

Definition at line 86 of file Vector3.H.

#### 10.324.3.6 **bool** Vector3::IsZero () const [inline]

comparison : zero

Definition at line 80 of file Vector3.H.

#### 10.324.3.7 **double** Vector3::Norm () const [inline]

eucliden norm

Definition at line 172 of file Vector3.H.

References sqrt().

#### 10.324.3.8 **bool** Vector3::Normalize () [inline]

Normalize

Definition at line 196 of file Vector3.H.

References sqrt().

**10.324.3.9** `double Vector3::NormSquared () const` [inline]

squared norm

Definition at line 178 of file Vector3.H.

**10.324.3.10** `Vector3 Vector3::operator * (double Scale) const` [inline]

operator \*(double)

Definition at line 116 of file Vector3.H.

References Vector3().

**10.324.3.11** `Vector3 Vector3::operator *= (double Scale)` [inline]

double multiply

Definition at line 153 of file Vector3.H.

**10.324.3.12** `bool Vector3::operator!= (const Vector3 & v) const` [inline]

comparison : not equal

Definition at line 74 of file Vector3.H.

**10.324.3.13** `Vector3 Vector3::operator+ (const Vector3 & v) const` [inline]

operator +

Definition at line 104 of file Vector3.H.

References Vector3(), x, y, and z.

**10.324.3.14** `Vector3 Vector3::operator+= (const Vector3 & v)` [inline]

operator +=

Definition at line 135 of file Vector3.H.

References x, y, and z.

**10.324.3.15** `Vector3 Vector3::operator- () const` [inline]

Unary minus

Definition at line 129 of file Vector3.H.

References Vector3().

#### 10.324.3.16 **Vector3** Vector3::operator- (const **Vector3** & v) const [inline]

operator -

Definition at line 110 of file Vector3.H.

References Vector3(), x, y, and z.

#### 10.324.3.17 **Vector3** Vector3::operator-= (const **Vector3** & v) [inline]

operator -=

Definition at line 144 of file Vector3.H.

References x, y, and z.

#### 10.324.3.18 **Vector3** Vector3::operator/ (double *Scale*) const [inline]

operator /(double)

Definition at line 122 of file Vector3.H.

References Vector3().

#### 10.324.3.19 **Vector3** Vector3::operator/= (double *Scale*) [inline]

double divide

Definition at line 162 of file Vector3.H.

#### 10.324.3.20 **Vector3&** Vector3::operator= (const **Vector3** & v) [inline]

copy operator

Definition at line 56 of file Vector3.H.

References x, y, and z.

#### 10.324.3.21 **bool** Vector3::operator== (const **Vector3** & v) const [inline]

comparison : equality

Definition at line 68 of file Vector3.H.

References x, y, and z.

**10.324.3.22 void Vector3::Set (double *a*, double *b*, double *c*) [inline]**

set

Definition at line 65 of file Vector3.H.

**10.324.4 Friends And Related Function Documentation****10.324.4.1 Vector3 operator \* (double *Scale*, const Vector3 & *v*) [friend]**

operator \*

Definition at line 98 of file Vector3.H.

**10.324.5 Member Data Documentation****10.324.5.1 double Vector3::x**

x,y,z coordinates, 0 at the initialisation

Definition at line 42 of file Vector3.H.

Referenced by Cross(), Dot(), GetNormalized(), IsNearlyEqual(), operator+(), operator+=(), operator-(), operator-=(), operator=(), operator==(), Matrix3::transform-Vector(), and Vector3().

**10.324.5.2 double Vector3::y**

x,y,z coordinates, 0 at the initialisation

Definition at line 42 of file Vector3.H.

Referenced by Cross(), Dot(), GetNormalized(), IsNearlyEqual(), operator+(), operator+=(), operator-(), operator-=(), operator=(), operator==(), Matrix3::transform-Vector(), and Vector3().

**10.324.5.3 double Vector3::z**

x,y,z coordinates, 0 at the initialisation

Definition at line 42 of file Vector3.H.

Referenced by Cross(), Dot(), GetNormalized(), IsNearlyEqual(), operator+(), operator+=(), operator-(), operator-=(), operator=(), operator==(), Matrix3::transform-Vector(), and Vector3().

The documentation for this class was generated from the following file:



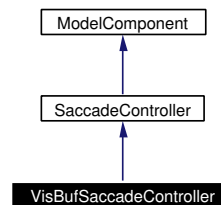
- Vector3.H

## 10.325 VisBufSaccadeController Class Reference

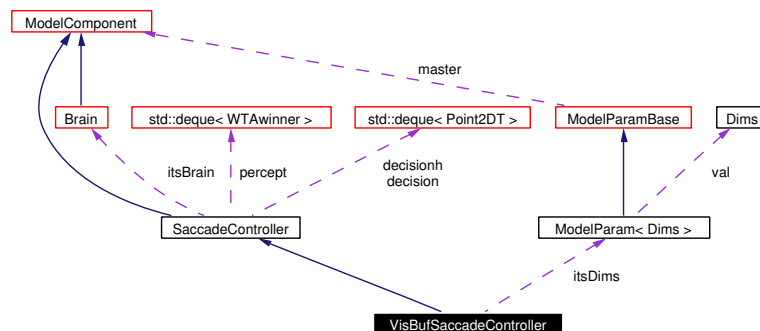
VisBuf saccade controller.

```
#include <SaccadeControllers.H>
```

Inheritance diagram for VisBufSaccadeController:



Collaboration diagram for VisBufSaccadeController:



### Public Member Functions

- [VisBufSaccadeController](#) ([ModelManager](#) &mgr)  
*Constructor.*
- virtual [~VisBufSaccadeController](#) ()  
*Destructor.*
- virtual void [exportOptions](#) (const int whatToExport, const bool recurse=true)  
*Export some of our [ModelParam](#) parameters as command-line-options.*
- virtual void [evolve](#) (const double t)

*evolve one time step (called, e.g., by [Brain::evolve\(\)](#))*

## Protected Member Functions

- virtual void [start1](#) ()  
*Get started.*
- virtual void [stop1](#) ()  
*get stopped*
- virtual void [computeWhenNewPercept](#) ()  
*This method is called each time a new percept has arrived.*
- virtual void [computeWhenResetPos](#) ()  
*This method is called each time a reset is made.*
- virtual [Point2D](#) [computeWhenNewDecision](#) (const double t)  
*This method is called each time a new decision is requested.*

## Protected Attributes

- [ModelParam](#)< [Dims](#) > [itsDims](#)  
*dimensions of the buffer*

### 10.325.1 Detailed Description

VisBuf saccade controller.

VisBufSaccadeController uses a world-centered [VisualBuffer](#) to accumulate interesting locations (each time a covert attention shift occurs and a percept is given to the controller), with internal competitive dynamics within the buffer.

Definition at line 737 of file SaccadeControllers.H.

### 10.325.2 Constructor & Destructor Documentation

#### 10.325.2.1 VisBufSaccadeController::VisBufSaccadeController ([ModelManager](#) & mgr)

Constructor.

Definition at line 1857 of file SaccadeControllers.C.

#### 10.325.2.2 **VisBufSaccadeController::~~VisBufSaccadeController ()** [virtual]

Destructor.

Definition at line 1865 of file SaccadeControllers.C.

### 10.325.3 Member Function Documentation

#### 10.325.3.1 **Point2D VisBufSaccadeController::computeWhenNewDecision** (const double t) [protected, virtual]

This method is called each time a new decision is requested.

Implements [SaccadeController](#).

Definition at line 1906 of file SaccadeControllers.C.

#### 10.325.3.2 **void VisBufSaccadeController::computeWhenNewPercept ()** [protected, virtual]

This method is called each time a new percept has arrived.

Implements [SaccadeController](#).

Definition at line 1897 of file SaccadeControllers.C.

#### 10.325.3.3 **void VisBufSaccadeController::computeWhenResetPos ()** [protected, virtual]

This method is called each time a reset is made.

Implements [SaccadeController](#).

Definition at line 1902 of file SaccadeControllers.C.

#### 10.325.3.4 **void VisBufSaccadeController::evolve (const double t) [virtual]**

evolve one time step (called, e.g., by [Brain::evolve\(\)](#))

Implements [SaccadeController](#).

Definition at line 1892 of file SaccadeControllers.C.

### 10.325.3.5 void VisBufSaccadeController::exportOptions (const int *whatToExport*, const bool *recurse* = true) [virtual]

Export some of our [ModelParam](#) parameters as command-line-options.

See the base function in [ModelComponent.H](#) for info.

Reimplemented from [ModelComponent](#).

Definition at line 1869 of file SaccadeControllers.C.

References [ModelComponent::exportOptions\(\)](#), [OPTEXP\\_CORE](#), and [ModelManager::requestOption\(\)](#).

### 10.325.3.6 void VisBufSaccadeController::start1 () [protected, virtual]

Get started.

Reimplemented from [ModelComponent](#).

Definition at line 1882 of file SaccadeControllers.C.

### 10.325.3.7 void VisBufSaccadeController::stop1 () [protected, virtual]

get stopped

Reimplemented from [ModelComponent](#).

Definition at line 1887 of file SaccadeControllers.C.

## 10.325.4 Member Data Documentation

### 10.325.4.1 [ModelParam<Dims>](#) VisBufSaccadeController::itsDims [protected]

dimensions of the buffer

Definition at line 755 of file SaccadeControllers.H.

The documentation for this class was generated from the following files:

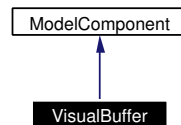
- [SaccadeControllers.H](#)
- [SaccadeControllers.C](#)

## 10.326 VisualBuffer Class Reference

An integrative visual buffer.

```
#include <VisualBuffer.H>
```

Inheritance diagram for VisualBuffer:



Collaboration diagram for VisualBuffer:

### Public Member Functions

- [VisualBuffer](#) ([ModelManager](#) &mgr, const [std::string](#) &descrName="Visual Buffer", const [std::string](#) &tagName="VisualBuffer")

*Constructor.*

- void [setBrain](#) ([Brain](#) \*brain)

*Set a hook to our [Brain](#).*

- [~VisualBuffer](#) ()

*Destructor.*

- virtual void [exportOptions](#) (const int whatToExport, const bool recurse=true)

*Export some of our [ModelParam](#) parameters as command-line-options.*

- void [input](#) (const [WTAwinner](#) &win, const [Image](#)< float > &sm, const [Image](#)< byte > &objmask)

*New input.*

- [Image](#)< byte > [getSaliencyMask](#) () const

*Get saliency mask.*

- [Image](#)< float > [getBuffer](#) (const bool normalize=false) const

*Get current buffer.*

- void [evolve](#) (const double t)

*evolve our internal dynamics*

- void [inhibit](#) (const [Point2D](#) &loc)  
*inhibit a location, given in retinotopic/retinal-scale coords*
- void [inhibit](#) (const [Image](#)< float > &mask)  
*inhibit a location, using a mask of same dims/scale as the buffer*
- [Point2D findMostInterestingTarget](#) (const [Point2D](#) &p)  
*Decide on most interesting target location given current eye position.*
- [Point2D findMostInterestingTargetLocMax](#) (const [Point2D](#) &p)  
*Decide on most interesting target location given current eye position.*
- [Point2D retinalToBuffer](#) (const [Point2D](#) &p) const  
*transform coord from retinotopic/retinal-scale to world-centered/sm-scale*
- [Point2D bufferToRetinal](#) (const [Point2D](#) &p) const  
*transform coord from world-centered/sm-scale to retinotopic/retinal-scale*
- bool [isObjectBased](#) () const  
*are we object-based?*

## Protected Member Functions

- void [start1](#) ()  
*get started (see [ModelComponent.H](#))*

## Protected Attributes

- [ModelParam](#)< bool > [itsIgnoreBoring](#)  
*ignore boring attention shifts*
- [ModelParam](#)< bool > [itsObjectBased](#)  
*true if doing object-based*
- [ModelParam](#)< [Dims](#) > [itsBufferDims](#)  
*dims of our internal buffer*
- [ModelParam](#)< [LevelSpec](#) > [itsLevelSpec](#)  
*levelspec determines buffer scale*

- [ModelParam< Dims > itsInputDims](#)  
*input image dims*
- [ModelParam< int > itsFOAradius](#)  
*radius of FOA*
- [ModelParam< double > itsTimePeriod](#)  
*period at which to apply interaction*
- [ModelParam< float > itsNormalizeFactor](#)  
*normalize factor for [getBuffer\(\)](#)*
- [ModelParam< float > itsDecayFactor](#)  
*temporal decay factor*
- [ModelParam< MaxNormType > itsNormType](#)  
*maxnorm type for our internals*

### 10.326.1 Detailed Description

An integrative visual buffer.

A world-centered Visual Buffer to accumulate interesting locations (each time a covert attention shift occurs), with internal competitive dynamics within the buffer.

Definition at line 52 of file VisualBuffer.H.

### 10.326.2 Constructor & Destructor Documentation

#### 10.326.2.1 VisualBuffer::VisualBuffer ([ModelManager](#) & mgr, const [std::string](#) & descrName = "Visual Buffer", const [std::string](#) & tagName = "VisualBuffer")

Constructor.

Definition at line 51 of file VisualBuffer.C.

References [VCXNORM\\_FANCY](#).

#### 10.326.2.2 VisualBuffer::~VisualBuffer ()

Destructor.



Definition at line 107 of file VisualBuffer.C.

### 10.326.3 Member Function Documentation

#### 10.326.3.1 [Point2D](#) VisualBuffer::bufferToRetinal (const [Point2D](#) & p) const

transform coord from world-centered/sm-scale to retinotopic/retinal-scale

Definition at line 308 of file VisualBuffer.C.

References [SharedPtr< T >::get\(\)](#), [Image< float >::getHeight\(\)](#), [Brain::getRET\(\)](#), [ModelParam< LevelSpec >::getVal\(\)](#), [Image< float >::getWidth\(\)](#), [Dims::h\(\)](#), [Point2D::i](#), [Point2D::j](#), [LevelSpec::mapLevel\(\)](#), and [Dims::w\(\)](#).

#### 10.326.3.2 void VisualBuffer::evolve (const double t)

evolve our internal dynamics

Definition at line 190 of file VisualBuffer.C.

References [ModelParam< double >::getVal\(\)](#), and [itsTimePeriod](#).

#### 10.326.3.3 void VisualBuffer::exportOptions (const int *whatToExport*, const bool *recurse* = true) [virtual]

Export some of our [ModelParam](#) parameters as command-line-options.

See the base function in [ModelComponent.H](#) for info.

Reimplemented from [ModelComponent](#).

Definition at line 69 of file VisualBuffer.C.

References [ModelComponent::exportOptions\(\)](#), [itsBufferDims](#), [itsDecayFactor](#), [itsIgnoreBoring](#), [itsNormalizeFactor](#), [itsObjectBased](#), [itsTimePeriod](#), [OPTEXP\\_CORE](#), and [ModelManager::requestOption\(\)](#).

#### 10.326.3.4 [Point2D](#) VisualBuffer::findMostInterestingTarget (const [Point2D](#) & p)

Decide on most interesting target location given current eye position.

All coordinates are in world-centered/sm-scale. The algorithm used here is inspired from the area activation model; we threshold out regions lower than a fraction of the maximum activation, then aim for the centroid of the region that is closest to current eye position.

Definition at line 216 of file VisualBuffer.C.

References `centroid()`, `chamfer34()`, `Image< T >::findMin()`, `Image< T >::flood()`, `Image< T >::getDims()`, `Image< T >::getMinMax()`, `makeBinary()`, `Image< T >::replaceVal()`, `Image< T >::setVal()`, and `ZEROS`.

#### 10.326.3.5 `Point2D` `VisualBuffer::findMostInterestingTargetLocMax (const Point2D & p)`

Decide on most interesting target location given current eye position.

All coordinates are in world-centered/sm-scale. The algorithm used here will aim for the local max above a threshold that is closest to current eye position.

Definition at line 244 of file `VisualBuffer.C`.

References `Point2D::distance()`, `Image< T >::getHeight()`, `Image< T >::getMinMax()`, `Image< T >::getVal()`, `Image< T >::getWidth()`, `Point2D::i`, `Point2D::j`, and `lowPass9()`.

#### 10.326.3.6 `Image< float >` `VisualBuffer::getBuffer (const bool normalize = false) const`

Get current buffer.

Definition at line 275 of file `VisualBuffer.C`.

References `ModelParam< float >::getVal()`, `itsNormalizeFactor`, and `Image< T >::normalize()`.

#### 10.326.3.7 `Image< byte >` `VisualBuffer::getSaliencyMask () const`

Get saliency mask.

This is mostly for display purposes. The mask is the one that was created at the last `input()` and has the same size/scale as the saliency map and is in retinal coordinates

Definition at line 186 of file `VisualBuffer.C`.

#### 10.326.3.8 `void VisualBuffer::inhibit (const Image< float > & mask)`

inhibit a location, using a mask of same dims/scale as the buffer

##### Parameters:

*mask* should have same dims as the buffer, and values between 0.0 (no inhibition) and 1.0 (full inhibition).

Definition at line 209 of file `VisualBuffer.C`.

References `binaryReverse()`.

### 10.326.3.9 void VisualBuffer::inhibit (const [Point2D](#) & *loc*)

inhibit a location, given in retinotopic/retinal-scale coords

A disk of radius `FOAradius` will be used for inhibition

Definition at line 199 of file `VisualBuffer.C`.

References `Image< T >::drawDisk()`, `Image< float >::getDims()`, `ModelParam< LevelSpec >::getVal()`, `ModelParam< int >::getVal()`, `LevelSpec::mapLevel()`, and `ZEROS`.

### 10.326.3.10 void VisualBuffer::input (const [WTAwinner](#) & *win*, const [Image](#)< float > & *sm*, const [Image](#)< byte > & *objmask*)

New input.

Beware that the inputs here are matched to what [Brain](#) naturally provides, which is

#### Parameters:

***win*** the coordinates of an attention shift at retinal resolution

***sm*** the saliency map at the maplevel resolution

***objmask*** a mask with (if initialized) the shape of the attended object at retinal resolution.

Definition at line 111 of file `VisualBuffer.C`.

References `binaryReverse()`, `WTAwinner::boring`, `chamfer34()`, `Image< T >::clear()`, `Image< T >::drawDisk()`, `Image< float >::getDims()`, `Image< T >::getDims()`, `ModelParam< int >::getVal()`, `ModelParam< Dims >::getVal()`, `ModelParam< bool >::getVal()`, `Dims::h()`, `Image< T >::initialized()`, `Image< float >::initialized()`, `itsIgnoreBoring`, `itsObjectBased`, `Image< T >::lowThresh()`, `WTAwinner::p`, `pasteImage()`, `rescale()`, `Image< T >::resize()`, `Image< float >::resize()`, `retinalToBuffer()`, `squash()`, `takeMax()`, `Dims::w()`, and `ZEROS`.

### 10.326.3.11 bool VisualBuffer::isObjectBased () const

are we object-based?

Definition at line 329 of file `VisualBuffer.C`.

References `ModelParam< bool >::getVal()`, and `itsObjectBased`.

**10.326.3.12 [Point2D](#) VisualBuffer::retinalToBuffer (const [Point2D](#) & *p*) const**

transform coord from retinotopic/retinal-scale to world-centered/sm-scale

Definition at line 285 of file VisualBuffer.C.

References [SharedPtr< T >::get\(\)](#), [Image< float >::getHeight\(\)](#), [Brain::getRET\(\)](#), [ModelParam< LevelSpec >::getVal\(\)](#), [Image< float >::getWidth\(\)](#), [Dims::h\(\)](#), [Point2D::i](#), [Point2D::j](#), [LevelSpec::mapLevel\(\)](#), and [Dims::w\(\)](#).

Referenced by [input\(\)](#).

**10.326.3.13 void VisualBuffer::setBrain ([Brain](#) \* *brain*)**

Set a hook to our [Brain](#).

Definition at line 103 of file VisualBuffer.C.

**10.326.3.14 void VisualBuffer::start1 () [protected, virtual]**

get started (see [ModelComponent.H](#))

Reimplemented from [ModelComponent](#).

Definition at line 90 of file VisualBuffer.C.

References [ModelParam< float >::getVal\(\)](#), [ModelParam< Dims >::getVal\(\)](#), [ModelParam< MaxNormType >::getValString\(\)](#), [Dims::h\(\)](#), [itsBufferDims](#), [itsDecayFactor](#), [Image< float >::resize\(\)](#), and [Dims::w\(\)](#).

**10.326.4 Member Data Documentation****10.326.4.1 [ModelParam<Dims>](#) VisualBuffer::itsBufferDims [protected]**

dims of our internal buffer

Definition at line 128 of file VisualBuffer.H.

Referenced by [exportOptions\(\)](#), and [start1\(\)](#).

**10.326.4.2 [ModelParam<float>](#) VisualBuffer::itsDecayFactor [protected]**

temporal decay factor

Definition at line 134 of file VisualBuffer.H.

Referenced by [exportOptions\(\)](#), and [start1\(\)](#).

**10.326.4.3   [ModelParam<int>](#)   [VisualBuffer::itsFOAradius](#)   [protected]**

radius of FOA

Definition at line 131 of file VisualBuffer.H.

**10.326.4.4   [ModelParam<bool>](#)   [VisualBuffer::itsIgnoreBoring](#)   [protected]**

ignore boring attention shifts

Definition at line 126 of file VisualBuffer.H.

Referenced by exportOptions(), and input().

**10.326.4.5   [ModelParam<Dims>](#)   [VisualBuffer::itsInputDims](#)   [protected]**

input image dims

Definition at line 130 of file VisualBuffer.H.

**10.326.4.6   [ModelParam<LevelSpec>](#)   [VisualBuffer::itsLevelSpec](#)   [protected]**

levelspec determines buffer scale

Definition at line 129 of file VisualBuffer.H.

**10.326.4.7   [ModelParam<float>](#)   [VisualBuffer::itsNormalizeFactor](#)   [protected]**

normalize factor for [getBuffer\(\)](#)

Definition at line 133 of file VisualBuffer.H.

Referenced by exportOptions(), and getBuffer().

**10.326.4.8   [ModelParam<MaxNormType>](#)   [VisualBuffer::itsNormType](#)   [protected]**

maxnorm type for our internals

Definition at line 135 of file VisualBuffer.H.

**10.326.4.9   [ModelParam<bool>](#)   [VisualBuffer::itsObjectBased](#)   [protected]**

true if doing object-based

Definition at line 127 of file VisualBuffer.H.

Referenced by exportOptions(), input(), and isObjectBased().

**10.326.4.10   [ModelParam<double>](#)   [VisualBuffer::itsTimePeriod](#)  
[protected]**

period at which to apply interaction

Definition at line 132 of file VisualBuffer.H.

Referenced by evolve(), and exportOptions().

The documentation for this class was generated from the following files:

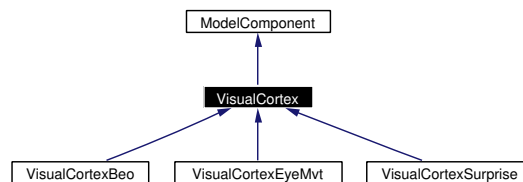
- [VisualBuffer.H](#)
- [VisualBuffer.C](#)

## 10.327 VisualCortex Class Reference

The Visual Cortex Class.

```
#include <VisualCortex.H>
```

Inheritance diagram for VisualCortex:



Collaboration diagram for VisualCortex:

### Public Member Functions

- const [Dims](#) & [getInputDims](#) () const  
*Returns the dimensions of the most recent input image.*

### Constructors/Destructor

- [VisualCortex](#) ([ModelManager](#) &mgr, const [std::string](#) &descrName="Visual Cortex", const [std::string](#) &tagName="VisualCortex")  
*Construct with no channel; channels must then be added with [addChannel\(\)](#).*
- [VisualCortex](#) ([ModelManager](#) &mgr, const [std::string](#) &descrName, const [std::string](#) &tagName, const double chanIw, const double chanCw, const double chanOw, const double chanFw, const double chanMw)  
*Construct with default intens/color/orient/flicker/motion channels.*
- [VisualCortex](#) ([ModelManager](#) &mgr, const [std::string](#) &descrName, const [std::string](#) &tagName, const double chanIw, const double chanCw, const double chanCow, const double chanOw, const double chanFw, const double chanMw, const double chanLw, const double chanTw, const double chanXw, const double chanEw, const double chanDw, const double chanSHw)
- virtual [~VisualCortex](#) ()  
*Virtual destructor for safe inheritance.*
- virtual void [exportOptions](#) (const int whatToExport, const bool recurse=true)

Export some of our [ModelParam](#) parameters as command-line-options.

- virtual void [reset](#) (const bool recurse=true)  
*Reset VisualCortex and propagate the reset signal to the sub-components.*

### Channel management functions

These functions offer an interface in which [VisualCortex](#) maintains a set of channels, each of which can be looked up either by an integer index or by an associated name.

- virtual [uint](#) [addChannel](#) (const [SharedPtr](#)< [ChannelBase](#) > &ch)  
*Adds a new channel to the VisualCortex.*
- virtual [uint](#) [addChannel](#) (const char \*name, const [SharedPtr](#)< [ChannelBase](#) > &ch)  
*Adds a new named channel to the VisualCortex.*
- virtual void [addDefaultChannels](#) (const double chanIw=1.0, const double chanCw=1.0, const double chanCow=1.0, const double chanOw=1.0, const double chanFw=1.0, const double chanMw=1.0, const double chanLw=0.0, const double chanTw=0.0, const double chanXw=0.0, const double chanEw=0.0, const double chanDw=0.0, const double chanSHw=0.0)  
*Adds the default channels to the VisualCortex object.*
- virtual [uint](#) [numChannels](#) () const  
*Get the number of contained channels.*
- virtual [ChannelBase](#) & [channel](#) (const [uint](#) idx) const  
*Look up a channel by its index (idx must be less than [numChannels\(\)](#)).*
- virtual [SharedPtr](#)< [ChannelBase](#) > [channelPtr](#) (const [uint](#) idx) const  
*Look up a channel by its index (idx must be less than [numChannels\(\)](#)).*
- virtual [SharedPtr](#)< [ChannelBase](#) > [channelPtr](#) (const char \*tagname) const  
*Look up a channel by its tagname.*
- virtual [ChannelBase](#) & [channel](#) (const char \*name) const  
*Look up a channel by its name.*
- virtual bool [hasChannel](#) (const char \*name) const  
*see whether we have a channel of given name*

### Processing functions



- virtual void **input** (const **Image**< **PixRGB**< **byte** > > &inp, const double t, const **Image**< **byte** > &clipMask=**Image**< **byte** >())  
*Present new input (e.g., a new frame from a video stream):.*
- virtual void **drawResults** (**Image**< **PixRGB**< **byte** > > &traj)  
*Call **drawResults()** for each contained channel.*
- virtual void **saveResults** (const **SharedPtr**< **OutputFrameSeries** > &ofs)  
*Call **saveResults()** for each contained channel.*
- virtual void **learn** (const **Image**< **byte** > &targets, const double &eta, const bool doDistMap=true, const bool softmax=false, const int inthresh=20, const int outthresh=255)  
*Learn feature map weights:.*
- virtual **Image**< float > **getJetDistMap** (const **Point2D** &center, const **Jet**< float > &weights)  
*get a Jet-distance map*
- virtual **Image**< float > **getRaoJetMap** (const **Jet**< float > &targ, const int smin, const int smax)  
*Get a jet distance map a la Rao et al., Vis Res, 2002.*

### Access functions

- virtual **Image**< float > **getOutput** ()  
*Get the output from visual cortex.*
- virtual **SharedPtr**< **JetSpec** > **getJetSpec** () const  
*get the **JetSpec** corresponding to our channels*
- virtual void **fillJet** (const **Point2D** &loc, **Jet**< float > &j) const  
*Fill the **Jet** with the relevant components from each contained channel.*
- virtual void **fillJetNI** (const **Point2D** &loc, **Jet**< float > &j) const  
*Fill the **Jet** with the relevant components from each contained channel.*

### Parameter management functions

*These functions generally just loop over the contained channels and forward the request to each of them.*

- void **clampParams** (const double &cmin, const double &cmax)  
*Parameter management function.*

- void [normalizeParams](#) ()  
*Normalize parameters to fixed sum.*
- void [cleanParams](#) (const double &epsilon)  
*Remove pars  $\leq$  epsilon for speedup of output computation:.*
- void [readFrom](#) (const [ParamMap](#) &pmap)  
*Read params from a [ParamMap](#).*
- void [writeTo](#) ([ParamMap](#) &pmap) const  
*Write params to a [ParamMap](#).*
- bool [readParamMap](#) (const char \*fname)  
*Read params from disk file.*
- void [writeParamMap](#) (const char \*fname) const  
*Write params to disk file.*
- void [biasFeatures](#) (const double \*mean, const double \*sigma)  
*Bias features( mean, sigma) of all featuremaps.*
- void [getFeatures](#) (const [Point2D](#) &locn, [std::vector](#)< double > &mean) const  
*Get the center-surround feature responses of all feature maps.*
- void [getFeatures](#) (const [Point2D](#) &locn, [std::vector](#)< double > &mean, int c\_start, int c\_end) const  
*Get the center-surround feature responses of all feature maps.*
- void [findWeight](#) ()  
*Find the weights of all channels after biasing features.*
- bool [reco](#) (const [ParamMap](#) &pmap, double &mismatch)  
*Recognise the location to check if it is the same as obj.*
- void [findLikelihood](#) (const [ParamMap](#) &pmap, double &likelihood)  
*what is the probability that the fixation belongs to this object?*

## Protected Attributes

- [ModelParam](#)< [MaxNormType](#) > [itsNormType](#)  
*maxNormalization to use*
- [ModelParam](#)< bool > [itsUseRandom](#)  
*add random noise to output*

- [ModelParam](#)< [std::string](#) > [itsBiasFeatures](#)  
*bias features with target params*
- [ModelParam](#)< [bool](#) > [itsSaveOutput](#)  
*save our output?*
- [ModelParam](#)< [float](#) > [itsOutputFactor](#)  
*output range factor*
- [ModelParam](#)< [bool](#) > [itsUseOlderVersion](#)
- [ModelParam](#)< [LevelSpec](#) > [itsLevelSpec](#)  
*LevelSpec used by our channels, used to compute output dims.*
- [ModelParam](#)< [float](#) > [itsSurpriseThresh](#)  
*surprise nonlinarity threshold*
- [ModelParam](#)< [float](#) > [itsSurpriseSemisat](#)  
*surprise nonlinarity semisat*
- [ModelParam](#)< [float](#) > [itsSurpriseExp](#)  
*surprise nonlinarity exponent*

### 10.327.1 Detailed Description

The Visual Cortex Class.

In brief, VisualCortex holds a collection of [ChannelBase](#) objects, and most of VisualCortex's operations are achieved by some kind of iteration over that collection. That is, VisualCortex now does little work by itself, but delegates its operations to the channels, accumulating their results if necessary.

Definition at line 73 of file VisualCortex.H.

### 10.327.2 Constructor & Destructor Documentation

**10.327.2.1** [VisualCortex::VisualCortex](#) ([ModelManager](#) & *mgr*, const [std::string](#) & *descrName* = "Visual Cortex", const [std::string](#) & *tagName* = "VisualCortex")

Construct with no channel; channels must then be added with [addChannel\(\)](#).

**Parameters:**

*mgr* our [ModelManager](#) (see [ModelManager.H](#))

*descrName* descriptive name for human useage

*tagName* name for [ParamMap](#) usage

Definition at line 58 of file VisualCortex.C.

References MaxNormType, and VCXNORM\_FANCY.

**10.327.2.2 VisualCortex::VisualCortex ([ModelManager](#) & *mgr*, const [std::string](#) & *descrName*, const [std::string](#) & *tagName*, const double *chanIw*, const double *chanCw*, const double *chanOw*, const double *chanFw*, const double *chanMw*)**

Construct with default intens/color/orient/flicker/motion channels.

This just calls [addDefaultChannels\(\)](#). The total channel weights are given here for each channel. If any is <= 0.0, the channel will NOT be added.

Definition at line 74 of file VisualCortex.C.

References addDefaultChannels(), MaxNormType, and VCXNORM\_FANCY.

**10.327.2.3 VisualCortex::VisualCortex ([ModelManager](#) & *mgr*, const [std::string](#) & *descrName*, const [std::string](#) & *tagName*, const double *chanIw*, const double *chanCw*, const double *chanCow*, const double *chanOw*, const double *chanFw*, const double *chanMw*, const double *chanLw*, const double *chanTw*, const double *chanXw*, const double *chanEw*, const double *chanDw*, const double *chanSHw*)**

This just calls [addDefaultChannels\(\)](#). The total channel weights are given here for each channel. If any is <= 0.0, the channel will NOT be added.

Definition at line 96 of file VisualCortex.C.

References addDefaultChannels(), MaxNormType, and VCXNORM\_FANCY.

**10.327.2.4 VisualCortex::~VisualCortex () [virtual]**

Virtual destructor for safe inheritance.

Definition at line 122 of file VisualCortex.C.

### 10.327.3 Member Function Documentation

#### 10.327.3.1 **uint VisualCortex::addChannel (const char \* *name*, const SharedPtr< ChannelBase > & *ch*) [virtual]**

Adds a new named channel to the VisualCortex.

The channel's tagName will be set to *name*. Returns the index at which the channel was inserted; this index could be passed back to [channel\(\)](#) to retrieve the channel object.

Definition at line 242 of file VisualCortex.C.

References [ModelComponent::addSubComponent\(\)](#), and [uint](#).

#### 10.327.3.2 **uint VisualCortex::addChannel (const SharedPtr< ChannelBase > & *ch*) [virtual]**

Adds a new channel to the VisualCortex.

The channel's tagName will be used for associative lookup. Returns the index at which the channel was inserted; this index could be passed back to [channel\(\)](#) to retrieve the channel object.

Definition at line 235 of file VisualCortex.C.

References [ModelComponent::addSubComponent\(\)](#), and [uint](#).

Referenced by [VisualCortexSurprise::addDefaultChannels\(\)](#), [VisualCortexBeo::addDefaultChannels\(\)](#), and [addDefaultChannels\(\)](#).

#### 10.327.3.3 **void VisualCortex::addDefaultChannels (const double *chanIw* = 1.0, const double *chanCw* = 1.0, const double *chanCow* = 1.0, const double *chanOw* = 1.0, const double *chanFw* = 1.0, const double *chanMw* = 1.0, const double *chanLw* = 0.0, const double *chanTw* = 0.0, const double *chanXw* = 0.0, const double *chanEw* = 0.0, const double *chanDw* = 0.0, const double *chanSHw* = 0.0) [virtual]**

Adds the default channels to the VisualCortex object.

intens/color/orient/flicker/motion/junctions/dummy channels. The total channel weights are given here for each channel. If any is <= 0.0, the channel will NOT be added.

Definition at line 158 of file VisualCortex.C.

References [addChannel\(\)](#), [channel\(\)](#), [makeShared\(\)](#), and [ChannelBase::setTotalWeight\(\)](#).

Referenced by [VisualCortex\(\)](#).

#### 10.327.3.4 **void VisualCortex::biasFeatures (const double \* *mean*, const double \* *sigma*)**

Bias features( mean, sigma) of all featuremaps.

Definition at line 686 of file VisualCortex.C.

References ChannelBase::biasFeatures(), channel(), clampParams(), findWeight(), numChannels(), and uint.

#### 10.327.3.5 **ChannelBase & VisualCortex::channel (const char \* *name*) const** [virtual]

Look up a channel by its name.

Definition at line 388 of file VisualCortex.C.

References ModelComponent::subComponent().

#### 10.327.3.6 **ChannelBase & VisualCortex::channel (const uint *idx*) const** [virtual]

Look up a channel by its index (*idx* must be less than [numChannels\(\)](#)).

Definition at line 364 of file VisualCortex.C.

References ModelComponent::subComponent(), and uint.

Referenced by VisualCortexSurprise::addDefaultChannels(), VisualCortexBeo::addDefaultChannels(), addDefaultChannels(), biasFeatures(), clampParams(), cleanParams(), JunctionChannel::doInput(), drawResults(), fillJet(), fillJetNI(), findLikelihood(), findWeight(), getFeatures(), getJetSpec(), getOutput(), input(), learn(), normalizeParams(), readFrom(), reco(), saveResults(), and writeTo().

#### 10.327.3.7 **SharedPtr< ChannelBase > VisualCortex::channelPtr (const char \* *tagname*) const** [virtual]

Look up a channel by its tagname.

Definition at line 368 of file VisualCortex.C.

References SharedPtr< T >::dynCastFrom(), SharedPtr< T >::get(), and ModelComponent::subComponent().

#### 10.327.3.8 **SharedPtr< ChannelBase > VisualCortex::channelPtr (const uint *idx*) const** [virtual]

Look up a channel by its index (*idx* must be less than [numChannels\(\)](#)).

Definition at line 378 of file VisualCortex.C.

References [SharedPtr< T >::dynCastFrom\(\)](#), [SharedPtr< T >::get\(\)](#), [ModelComponent::subComponent\(\)](#), and [uint](#).

#### 10.327.3.9 void VisualCortex::clampParams (const double & *cmin*, const double & *cmax*)

Parameter management function.

Definition at line 515 of file VisualCortex.C.

References [channel\(\)](#), [ChannelBase::clampCoeffs\(\)](#), [ChannelBase::clampWeight\(\)](#), [numChannels\(\)](#), and [uint](#).

Referenced by [biasFeatures\(\)](#).

#### 10.327.3.10 void VisualCortex::cleanParams (const double & *epsilon*)

Remove pars <= epsilon for speedup of output computation:.

Definition at line 549 of file VisualCortex.C.

References [channel\(\)](#), [ChannelBase::cleanCoeffs\(\)](#), [ChannelBase::cleanWeight\(\)](#), [numChannels\(\)](#), and [uint](#).

#### 10.327.3.11 void VisualCortex::drawResults ([Image< PixRGB< byte > > & traj](#)) [virtual]

Call [drawResults\(\)](#) for each contained channel.

Definition at line 325 of file VisualCortex.C.

References [channel\(\)](#), [ChannelBase::drawResults\(\)](#), [numChannels\(\)](#), and [uint](#).

#### 10.327.3.12 void VisualCortex::exportOptions (const int *whatToExport*, const bool *recurse* = true) [virtual]

Export some of our [ModelParam](#) parameters as command-line-options.

See the base function in [ModelComponent.H](#) for info.

Reimplemented from [ModelComponent](#).

Reimplemented in [VisualCortexEyeMvt](#).

Definition at line 126 of file VisualCortex.C.

References `ModelComponent::exportOptions()`, `itsBiasFeatures`, `itsOutputFactor`, `itsSurpriseExp`, `itsSurpriseSemisat`, `itsSurpriseThresh`, `OPTEXP_CORE`, `OPTEXP_SAVE`, and `ModelManager::requestOption()`.

Referenced by `VisualCortexEyeMvt::exportOptions()`.

**10.327.3.13 void VisualCortex::fillJet (const [Point2D](#) & loc, [Jet](#)< float > & j)  
const [virtual]**

Fill the [Jet](#) with the relevant components from each contained channel.

Definition at line 501 of file `VisualCortex.C`.

References `channel()`, `ChannelBase::fillJet()`, `numChannels()`, and `uint`.

Referenced by `getJetDistMap()`.

**10.327.3.14 void VisualCortex::fillJetNI (const [Point2D](#) & loc, [Jet](#)< float > & j)  
const [virtual]**

Fill the [Jet](#) with the relevant components from each contained channel.

this version does not use interpolation

Definition at line 508 of file `VisualCortex.C`.

References `channel()`, `ChannelBase::fillJetNI()`, `numChannels()`, and `uint`.

Referenced by `getRaoJetMap()`.

**10.327.3.15 void VisualCortex::findLikelihood (const [ParamMap](#) & pmap,  
double & likelihood)**

what is the probability that the fixation belongs to this object?

Definition at line 740 of file `VisualCortex.C`.

References `channel()`, `ChannelBase::findLikelihood()`, `ParamMap::getSubmap()`, `ParamMap::hasParam()`, `numChannels()`, `ModelComponent::tagName()`, and `uint`.

**10.327.3.16 void VisualCortex::findWeight ()**

Find the weights of all channels after biasing features.

Definition at line 700 of file `VisualCortex.C`.

References `channel()`, `ChannelBase::findWeight()`, `ChannelBase::getTotalWeight()`, `numChannels()`, `ChannelBase::setTotalWeight()`, and `uint`.

Referenced by `biasFeatures()`.



**10.327.3.17** `void VisualCortex::getFeatures (const Point2D & locn, std::vector< double > & mean, int c_start, int c_end) const`

Get the center-surround feature responses of all feature maps.

this will only return values for channles between *c\_start* and *c\_end*

Definition at line 673 of file VisualCortex.C.

References ASSERT, channel(), ChannelBase::getFeatures(), and numChannels().

**10.327.3.18** `void VisualCortex::getFeatures (const Point2D & locn, std::vector< double > & mean) const`

Get the center-surround feature responses of all feature maps.

Definition at line 663 of file VisualCortex.C.

References channel(), ChannelBase::getFeatures(), numChannels(), and uint.

**10.327.3.19** `const Dims & VisualCortex::getInputDims () const`

Returns the dimensions of the most recent input image.

Definition at line 615 of file VisualCortex.C.

**10.327.3.20** `Image< float > VisualCortex::getJetDistMap (const Point2D & center, const Jet< float > & weights) [virtual]`

get a Jet-distance map

#### Parameters:

***center*** the location at which the reference [Jet](#) will be extracted, using the [JetSpec](#) that is underling in the weights

***weights*** the weights to be used when computing a weighted RMS distance between the [Jet](#) extractred at the center location and those extracted at ev-eryother location in the image. This distance will be stored in the returned image for every pixel.

Definition at line 619 of file VisualCortex.C.

References distance(), fillJet(), Jet< T >::getSpec(), Dims::h(), NO\_INIT, Image< T >::setVal(), and Dims::w().

### 10.327.3.21 `SharedPtr< JetSpec > VisualCortex::getJetSpec () const` [virtual]

get the `JetSpec` corresponding to our channels

Definition at line 493 of file `VisualCortex.C`.

References `channel()`, `ChannelBase::fillJetSpec()`, `numChannels()`, and `uint`.

### 10.327.3.22 `Image< float > VisualCortex::getOutput ()` [virtual]

Get the output from visual cortex.

This will be some kind of normalized sum of the outputs of each of the contained channels.

Reimplemented in `VisualCortexEyeMvt`.

Definition at line 396 of file `VisualCortex.C`.

References `Image< float >::addBGnoise()`, `channel()`, `ModelManager::debug-Mode()`, `downSizeClean()`, `Image< float >::getDims()`, `Image< float >::get-MinMax()`, `ChannelBase::getOutput()`, `ModelParam< float >::getVal()`, `Model-Param< MaxNormType >::getVal()`, `ModelParam< LevelSpec >::getVal()`, `Model-Param< bool >::getVal()`, `Dims::h()`, `Image< float >::initialized()`, `ChannelBase::is-ZeroWeight()`, `itsOutputFactor`, `itsSurpriseExp`, `itsSurpriseSemisat`, `itsSurprise-Thresh`, `LevelSpec::mapLevel()`, `maxNormalize()`, `maxNormTypeName()`, `num-Channels()`, `ChannelBase::outputAvailable()`, `Image< float >::rectify()`, `Image< float >::sigmoid()`, `ModelComponent::tagName()`, `uint`, `VCXNORM_LANDMARK`, `VCXNORM_NONE`, `VCXNORM_SURPRISE`, `Dims::w()`, and `ZEROS`.

Referenced by `saveResults()`.

### 10.327.3.23 `Image< float > VisualCortex::getRaoJetMap (const Jet< float > & targ, const int smin, const int smax)` [virtual]

Get a jet distance map a la Rao et al., Vis Res, 2002.

Definition at line 643 of file `VisualCortex.C`.

References `fillJetNI()`, `Jet< T >::getSpec()`, `Dims::h()`, `NO_INIT`, `raodistance()`, `Image< T >::setVal()`, and `Dims::w()`.

### 10.327.3.24 `bool VisualCortex::hasChannel (const char * name) const` [virtual]

see whether we have a channel of given name

Definition at line 392 of file `VisualCortex.C`.

References `ModelComponent::hasSubComponent()`.

Referenced by `EndPointChannel::doInput()`, `XJunctionChannel::doInput()`, `TJunctionChannel::doInput()`, `LJunctionChannel::doInput()`, and `JunctionChannel::doInput()`.

**10.327.3.25** `void VisualCortex::input (const Image< PixRGB< byte > > & inp, const double t, const Image< byte > & clipMask = Image< byte >()) [virtual]`

Present new input (e.g., a new frame from a video stream):.

Reimplemented in [VisualCortexEyeMvt](#).

Definition at line 253 of file `VisualCortex.C`.

References `channel()`, `featureHierarchyLevel()`, `Image< float >::freeMem()`, `ChannelBase::input()`, `input()`, `itsBiasFeatures`, `luminance()`, `numChannels()`, `readParamMap()`, `ModelComponent::tagName()`, and `uint`.

Referenced by `input()`.

**10.327.3.26** `void VisualCortex::learn (const Image< byte > & targets, const double & eta, const bool doDistMap = true, const bool softmax = false, const int inthresh = 20, const int outthresh = 255) [virtual]`

Learn feature map weights:.

CAUTION: targets should correspond to last loaded input!

Definition at line 340 of file `VisualCortex.C`.

References `binaryReverse()`, `byte`, `chamfer34()`, `channel()`, `ChannelBase::learn()`, `numChannels()`, and `uint`.

**10.327.3.27** `void VisualCortex::normalizeParams ()`

Normalize parameters to fixed sum.

Good thing to do after learning to prevent blowup.

Definition at line 525 of file `VisualCortex.C`.

References `ChannelBase::absSumCoeffs()`, `channel()`, `ChannelBase::getTotalWeight()`, `ChannelBase::normalizeCoeffs()`, `numChannels()`, `ChannelBase::numSubmaps()`, and `uint`.

**10.327.3.28    [uint](#) VisualCortex::numChannels () const    [virtual]**

Get the number of contained channels.

Definition at line 360 of file VisualCortex.C.

References [ModelComponent::numSubComp\(\)](#), and [uint](#).

Referenced by [biasFeatures\(\)](#), [clampParams\(\)](#), [cleanParams\(\)](#), [drawResults\(\)](#), [fillJet\(\)](#), [fillJetNI\(\)](#), [findLikelihood\(\)](#), [findWeight\(\)](#), [getFeatures\(\)](#), [getJetSpec\(\)](#), [getOutput\(\)](#), [input\(\)](#), [learn\(\)](#), [normalizeParams\(\)](#), [readFrom\(\)](#), [reco\(\)](#), [saveResults\(\)](#), and [writeTo\(\)](#).

**10.327.3.29    void VisualCortex::readFrom (const [ParamMap](#) & *pmap*)**

Read params from a [ParamMap](#).

This is used internally by [readParamMap\(\)](#). This is different from [ModelComponent::readParamsFrom\(\)](#) as it concerns mostly feature map weights rather than the various [ModelComponent](#) parameters.

Definition at line 590 of file VisualCortex.C.

References [channel\(\)](#), [ParamMap::getSubmap\(\)](#), [ParamMap::hasParam\(\)](#), [numChannels\(\)](#), [ChannelBase::readFrom\(\)](#), [ModelComponent::tagName\(\)](#), and [uint](#).

Referenced by [readParamMap\(\)](#).

**10.327.3.30    bool VisualCortex::readParamMap (const char \* *fname*)**

Read params from disk file.

Definition at line 559 of file VisualCortex.C.

References [ParamMap::load\(\)](#), and [readFrom\(\)](#).

Referenced by [input\(\)](#).

**10.327.3.31    bool VisualCortex::reco (const [ParamMap](#) & *pmap*, double & *mismatch*)**

Recognise the location to check if it is the same as obj.

Definition at line 722 of file VisualCortex.C.

References [channel\(\)](#), [ParamMap::getSubmap\(\)](#), [ParamMap::hasParam\(\)](#), [numChannels\(\)](#), [ChannelBase::numSubmaps\(\)](#), [ChannelBase::reco\(\)](#), [sqrt\(\)](#), [ModelComponent::tagName\(\)](#), and [uint](#).

**10.327.3.32 void VisualCortex::reset (const bool *recurse* = true) [virtual]**

Reset VisualCortex and propagate the reset signal to the sub-components.

See the base function in [ModelComponent.H](#) for info.

Reimplemented from [ModelComponent](#).

Reimplemented in [VisualCortexEyeMvt](#).

Definition at line 148 of file VisualCortex.C.

References `Image< float >::freeMem()`, and `ModelComponent::reset()`.

Referenced by `VisualCortexEyeMvt::reset()`.

**10.327.3.33 void VisualCortex::saveResults (const [SharedPtr< OutputFrameSeries >](#) & ofs) [virtual]**

Call [saveResults\(\)](#) for each contained channel.

Our output will be saved with prefix "VCO<tagname>-"

Definition at line 329 of file VisualCortex.C.

References `channel()`, `getOutput()`, `ModelParam< bool >::getVal()`, `numChannels()`, `ChannelBase::saveResults()`, and `uint`.

**10.327.3.34 void VisualCortex::writeParamMap (const char \* *fname*) const**

Write params to disk file.

Definition at line 575 of file VisualCortex.C.

References `ParamMap::format()`, and `writeTo()`.

**10.327.3.35 void VisualCortex::writeTo ([ParamMap](#) & *pmap*) const**

Write params to a [ParamMap](#).

This is used internally by [writeParamMap\(\)](#). This is different from [ModelComponent::writeParamsTo\(\)](#) as it concerns mostly feature map weights rather than the various [ModelComponent](#) parameters.

Definition at line 604 of file VisualCortex.C.

References `channel()`, `numChannels()`, `ParamMap::putSubmap()`, `ModelComponent::tagName()`, `uint`, and `ChannelBase::writeTo()`.

Referenced by `writeParamMap()`.

## 10.327.4 Member Data Documentation

### 10.327.4.1 [ModelParam<std::string>](#) [VisualCortex::itsBiasFeatures](#) [protected]

bias features with target params

Definition at line 306 of file VisualCortex.H.

Referenced by exportOptions(), and input().

### 10.327.4.2 [ModelParam<LevelSpec>](#) [VisualCortex::itsLevelSpec](#) [protected]

[LevelSpec](#) used by our channels, used to compute output dims.

Definition at line 332 of file VisualCortex.H.

### 10.327.4.3 [ModelParam<MaxNormType>](#) [VisualCortex::itsNormType](#) [protected]

maxNormalization to use

Definition at line 304 of file VisualCortex.H.

### 10.327.4.4 [ModelParam<float>](#) [VisualCortex::itsOutputFactor](#) [protected]

output range factor

Definition at line 308 of file VisualCortex.H.

Referenced by exportOptions(), and getOutput().

### 10.327.4.5 [ModelParam<bool>](#) [VisualCortex::itsSaveOutput](#) [protected]

save our output?

Definition at line 307 of file VisualCortex.H.

### 10.327.4.6 [ModelParam<float>](#) [VisualCortex::itsSurpriseExp](#) [protected]

surprise nonlinarity exponent

Definition at line 336 of file VisualCortex.H.

Referenced by `exportOptions()`, and `getOutput()`.

#### 10.327.4.7 **ModelParam**<float> **VisualCortex::itsSurpriseSemisat** [protected]

surprise nonlinarity semisat

Definition at line 335 of file `VisualCortex.H`.

Referenced by `exportOptions()`, and `getOutput()`.

#### 10.327.4.8 **ModelParam**<float> **VisualCortex::itsSurpriseThresh** [protected]

surprise nonlinarity threshold

Definition at line 334 of file `VisualCortex.H`.

Referenced by `exportOptions()`, and `getOutput()`.

#### 10.327.4.9 **ModelParam**<bool> **VisualCortex::itsUseRandom** [protected]

add random noise to output

Definition at line 305 of file `VisualCortex.H`.

The documentation for this class was generated from the following files:

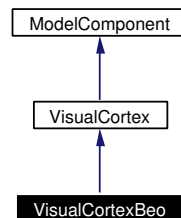
- [VisualCortex.H](#)
- [VisualCortex.C](#)

## 10.328 VisualCortexBeo Class Reference

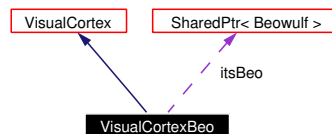
This is a derivation of [VisualCortex](#) that uses [SingleChannelBeo](#) channels.

```
#include <VisualCortexBeo.H>
```

Inheritance diagram for VisualCortexBeo:



Collaboration diagram for VisualCortexBeo:



### Public Member Functions

- [VisualCortexBeo](#) ([ModelManager](#) &mgr, const [std::string](#) &descrName, const [std::string](#) &tagName, const [SharedPtr](#)< [Beowulf](#) > &beo)

*Constructor without any channels.*

- [VisualCortexBeo](#) ([ModelManager](#) &mgr, const [std::string](#) &descrName, const [std::string](#) &tagName, const [SharedPtr](#)< [Beowulf](#) > &beo, const double chanIw, const double chanCw, const double chanOw, const double chanFw, const double chanMw)

*Construct with default intens/color/orient/flicker/motion beo channels.*

- virtual [~VisualCortexBeo](#) ()

*Virtual destructor for safe inheritance.*

- void [addDefaultChannels](#) (const double chanIw=1.0, const double chanCw=1.0, const double chanOw=1.0, const double chanFw=1.0, const double chanMw=1.0)



*Adds the default channels to the VisualCortexBeo object.*

- `SharedPtr< Beowulf > getBeowulf () const`  
*get our Beowulf*

## Protected Attributes

- `SharedPtr< Beowulf > itsBeo`  
*Our Beowulf (we are a Beowulf master).*

### 10.328.1 Detailed Description

This is a derivation of `VisualCortex` that uses `SingleChannelBeo` channels.

The only difference with `VisualCortex` is in `addDefaultChannels`, which will use the Beo version of the channels rather than the normal one. See `SingleChannelBeo.H` and `ChannelsBeo.H` for info. At construction, we require that a `Beowulf` initialized as master is given. You may still add whatever channels you want to this `VisualCortex`; just if you want them to be Beo too, make sure you first get our `Beowulf` using `getBeowulf()`, the next available node using `getNextAvailableNode()` and then initialize your `BeoChannels` using this `Beowulf` and node number.

If you are going to use a `VisualCortexBeo`, you should be running a process that implements a `SingleChannelBeoServer` on each slave node of your `Beowulf`. Such process is in `beochannel-server.C` for example. First start all your servers on all your server nodes, then, on the master, start your program that uses the `VisualCortexBeo`. It will configure the `Beowulf`, spawn the channels, and rip through your frames!

Definition at line 63 of file `VisualCortexBeo.H`.

### 10.328.2 Constructor & Destructor Documentation

#### 10.328.2.1 `VisualCortexBeo::VisualCortexBeo (ModelManager & mgr, const std::string & descrName, const std::string & tagName, const SharedPtr< Beowulf > & beo)`

Constructor without any channels.

Definition at line 43 of file `VisualCortexBeo.C`.

**10.328.2.2 VisualCortexBeo::VisualCortexBeo** ([ModelManager](#) & mgr, const [std::string](#) & descrName, const [std::string](#) & tagName, const [SharedPtr](#)< [Beowulf](#) > & beo, const double chanIw, const double chanCw, const double chanOw, const double chanFw, const double chanMw)

Construct with default intens/color/orient/flicker/motion beo channels.

This just calls [addDefaultChannels\(\)](#). The total channel weights are given here for each channel. If any is <= 0.0, the channel will NOT be added.

Definition at line 56 of file VisualCortexBeo.C.

References [addDefaultChannels\(\)](#).

**10.328.2.3 VisualCortexBeo::~~VisualCortexBeo** () [virtual]

Virtual destructor for safe inheritance.

Definition at line 75 of file VisualCortexBeo.C.

### 10.328.3 Member Function Documentation

**10.328.3.1 void VisualCortexBeo::addDefaultChannels** (const double chanIw = 1.0, const double chanCw = 1.0, const double chanOw = 1.0, const double chanFw = 1.0, const double chanMw = 1.0)

Adds the default channels to the VisualCortexBeo object.

intens/color/orient/flicker/motion beo channels. The total channel weights are given here for each channel. If any is <= 0.0, the channel will NOT be added.

Definition at line 79 of file VisualCortexBeo.C.

References [VisualCortex::addChannel\(\)](#), [VisualCortex::channel\(\)](#), [makeShared\(\)](#), and [ChannelBase::setTotalWeight\(\)](#).

Referenced by [VisualCortexBeo\(\)](#).

**10.328.3.2 [SharedPtr](#)< [Beowulf](#) > VisualCortexBeo::getBeowulf** () const

get our [Beowulf](#)

Definition at line 121 of file VisualCortexBeo.C.

## 10.328.4 Member Data Documentation

### 10.328.4.1 [SharedPtr<Beowulf>](#) [VisualCortexBeo::itsBeo](#) [protected]

Our [Beowulf](#) (we are a [Beowulf](#) master).

Definition at line 97 of file VisualCortexBeo.H.

The documentation for this class was generated from the following files:

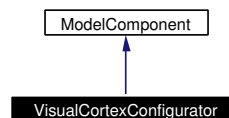
- [VisualCortexBeo.H](#)
- [VisualCortexBeo.C](#)

## 10.329 VisualCortexConfigurator Class Reference

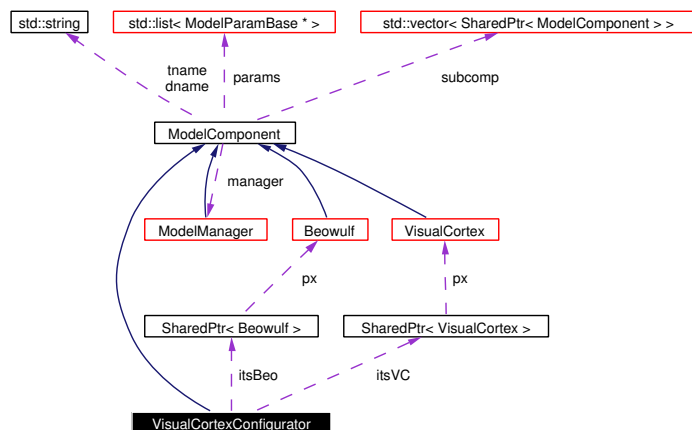
[VisualCortex](#) configurator.

```
#include <VisualCortexConfigurator.H>
```

Inheritance diagram for VisualCortexConfigurator:



Collaboration diagram for VisualCortexConfigurator:



### Public Member Functions

- [VisualCortexConfigurator](#) ([ModelManager](#) &mgr, const [std::string](#) &descr-Name="Visual Cortex Configurator", const [std::string](#) &tagName="Visual-CortexConfigurator")

*Constructor.*

- virtual [~VisualCortexConfigurator](#) ()

*destructor*

- virtual void [exportOptions](#) (const int whatToExport, const bool recurse=true)

*Export some of our [ModelParam](#) parameters as command-line-options.*

- `SharedPtr< VisualCortex > getVC () const`  
*Get the chosen VisualCortex.*
- `SharedPtr< Beowulf > getBeo () const`  
*If the chosen VisualCortex uses a Beowulf, get it here.*

## Protected Member Functions

- virtual bool `setModelParamString` (const `std::string` &name, const `std::string` &value, const bool recurse=false, const bool warn=true)  
*Intercept people changing our ModelParam.*

## Protected Attributes

- `ModelParam< std::string > itsVCtype`  
*type of visual cortex*

### 10.329.1 Detailed Description

`VisualCortex` configurator.

This will export the `-vc-type=XX` command-line option and will instantiate a `VisualCortex` of the desired type as the option gets assigned a value. As this happens, new options may become available in the command-line. To see them, use `-help` AFTER you have chosen the type to use. The current `VisualCortex` may be retrieved using `getVC()`.

Definition at line 52 of file `VisualCortexConfigurator.H`.

### 10.329.2 Constructor & Destructor Documentation

- 10.329.2.1 `VisualCortexConfigurator::VisualCortexConfigurator`  
(`ModelManager` & mgr, const `std::string` & descrName =  
"Visual Cortex Configurator", const `std::string` & tagName =  
"VisualCortexConfigurator")

Constructor.

Definition at line 52 of file `VisualCortexConfigurator.C`.

References OPTEXP\_NONE.

### 10.329.2.2 VisualCortexConfigurator::~~VisualCortexConfigurator () [virtual]

destructor

Definition at line 61 of file VisualCortexConfigurator.C.

## 10.329.3 Member Function Documentation

### 10.329.3.1 void VisualCortexConfigurator::exportOptions (const int *whatToExport*, const bool *recurse* = true) [virtual]

Export some of our [ModelParam](#) parameters as command-line-options.

See the base function in [ModelComponent.H](#) for info. Actually, here we just keep a copy of *whatToExport* and we will call [exportOptions\(\)](#) on our [VisualCortex](#) once we know its type and instantiate it.

Reimplemented from [ModelComponent](#).

Definition at line 65 of file VisualCortexConfigurator.C.

References [ModelComponent::exportOptions\(\)](#), [itsVCTYPE](#), OPTEXP\_CORE, and [ModelManager::requestOption\(\)](#).

### 10.329.3.2 SharedPtr< Beowulf > VisualCortexConfigurator::getBeo () const

If the chosen [VisualCortex](#) uses a [Beowulf](#), get it here.

You should call this during [start\(\)](#) of the [ModelComponent](#) that needs the [VisualCortex](#). CAUTION: this may be a [SharedPtr\(NULL\)](#) if no controller has been chosen.

Definition at line 85 of file VisualCortexConfigurator.C.

### 10.329.3.3 SharedPtr< VisualCortex > VisualCortexConfigurator::getVC () const

Get the chosen [VisualCortex](#).

You should call this during [start\(\)](#) of the [ModelComponent](#) that needs the [VisualCortex](#). CAUTION: this may be a [SharedPtr\(NULL\)](#) if no controller has been chosen.

Definition at line 81 of file VisualCortexConfigurator.C.

**10.329.3.4** `bool VisualCortexConfigurator::setModelParamString (const std::string & name, const std::string & value, const bool recurse = false, const bool warn = true)` [protected, virtual]

Intercept people changing our [ModelParam](#).

See [ModelComponent.H](#); as parsing the command-line or reading a config file sets our name, we'll also here instantiate a controller of the proper type (and export its options)

Reimplemented from [ModelComponent](#).

Definition at line 90 of file VisualCortexConfigurator.C.

References [ModelComponent::addSubComponent\(\)](#), [SharedPtr< Beowulf >::get\(\)](#), [SharedPtr< VisualCortex >::get\(\)](#), [itsVCtype](#), [ModelComponent::removeSubComponent\(\)](#), [SharedPtr< Beowulf >::reset\(\)](#), [SharedPtr< VisualCortex >::reset\(\)](#), [ModelComponent::setModelParamString\(\)](#), and [uint](#).

## 10.329.4 Member Data Documentation

**10.329.4.1** `ModelParam<std::string> VisualCortexConfigurator::itsVCtype`  
[protected]

type of visual cortex

Definition at line 86 of file VisualCortexConfigurator.H.

Referenced by [exportOptions\(\)](#), and [setModelParamString\(\)](#).

The documentation for this class was generated from the following files:

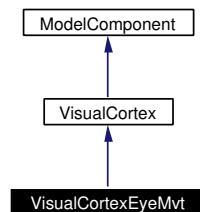
- [VisualCortexConfigurator.H](#)
- [VisualCortexConfigurator.C](#)

## 10.330 VisualCortexEyeMvt Class Reference

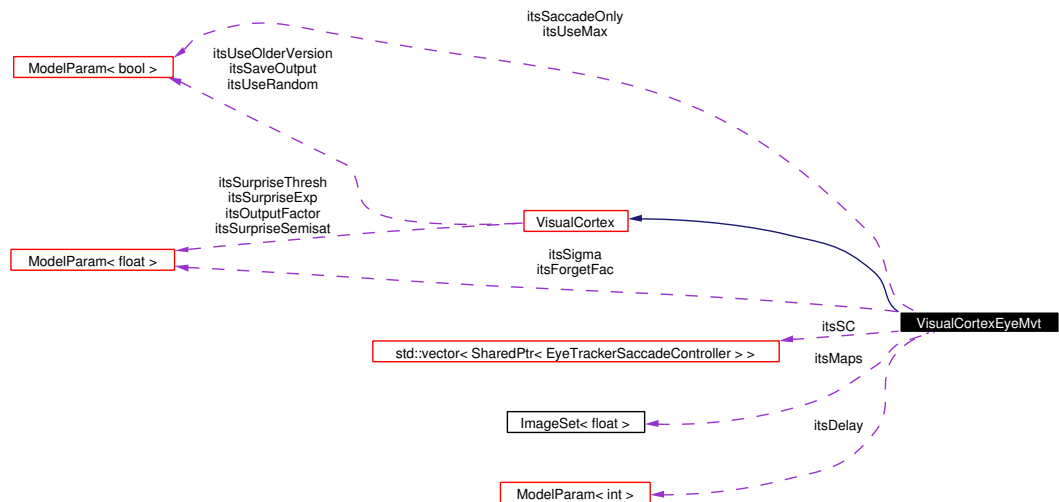
The VisualCortexEyeMvt Class.

```
#include <VisualCortexEyeMvt.H>
```

Inheritance diagram for VisualCortexEyeMvt:



Collaboration diagram for VisualCortexEyeMvt:



### Public Member Functions

- **VisualCortexEyeMvt** (**ModelManager** &mgr, const **std::string** &descr-Name="Visual Cortex Eye Mvt", const **std::string** &tagName="Visual-CortexEyeMvt")

*Construct with no channel; channels are ignored anyways.*



- [VisualCortexEyeMvt](#) ([ModelManager](#) &mgr, const [std::string](#) &descrName, const [std::string](#) &tagName, const double chanIw, const double chanCw, const double chanOw, const double chanFw, const double chanMw)  
*Construct, ignoring channel definitions.*
- [VisualCortexEyeMvt](#) ([ModelManager](#) &mgr, const [std::string](#) &descrName, const [std::string](#) &tagName, const double chanIw, const double chanCw, const double chanCow, const double chanOw, const double chanFw, const double chanMw, const double chanLw, const double chanTw, const double chanXw, const double chanEw, const double chanDw)  
*Construct, ignoring even more channel definitions.*
- virtual [~VisualCortexEyeMvt](#) ()  
*Virtual destructor for safe inheritance.*
- virtual void [exportOptions](#) (const int whatToExport, const bool recurse=true)  
*Export some of our [ModelParam](#) parameters as command-line-options.*
- virtual void [reset](#) (const bool recurse=true)  
*Reset [VisualCortex](#) and propagate the reset signal to the sub-components.*
- virtual void [input](#) (const [Image](#)< [PixRGB](#)< [byte](#) > > &inp, const double t, const [Image](#)< [byte](#) > &clipMask=[Image](#)< [byte](#) >())  
*Present new input (e.g., a new frame from a video stream):.*
- virtual [Image](#)< float > [getOutput](#) ()  
*Get the output from visual cortex.*
- virtual void [evolve](#) (const double stime)  
*evolve...*
- [uint](#) [numHumans](#) () const  
*Get our number of humans.*
- [Point2D](#) [getEyePos](#) (const [uint](#) human) const  
*Get current eye position of our of our humans.*

## Protected Member Functions

- virtual void [start1](#) ()  
*This is called from within [start\(\)](#) before the subcomponents start.*

## Protected Attributes

- [ModelParam](#)< [std::string](#) > [itsFNames](#)  
*names of eye movement files*
- [ModelParam](#)< [float](#) > [itsSigma](#)  
*sigma of gaussian blobs in pix at sml*
- [ModelParam](#)< [float](#) > [itsForgetFac](#)  
*forgetting factor*
- [ModelParam](#)< [int](#) > [itsDelay](#)  
*human-to-human delay, in eye samples*
- [ModelParam](#)< [bool](#) > [itsUseMax](#)  
*use max to combine instead of sum*
- [ModelParam](#)< [bool](#) > [itsSaccadeOnly](#)  
*plot only when a saccade*

### 10.330.1 Detailed Description

The VisualCortexEyeMvt Class.

This is a fake [VisualCortex](#) that outputs a saliency map built from a number of human eye movement traces. It contains a bunch of [EyeTrackerSaccadeController](#) objects that will provide eye position of the various humans, and builds some sort of heat map from those. Contrary to other [VisualCortex](#) variants, this requires an [evolve\(\)](#). Since [Brain](#) usually will not call [evolve\(\)](#) on its [VisualCortex](#), a special hook has been setup to handle that.

Definition at line 55 of file VisualCortexEyeMvt.H.

### 10.330.2 Constructor & Destructor Documentation

**10.330.2.1 VisualCortexEyeMvt::VisualCortexEyeMvt ([ModelManager](#) & mgr, const [std::string](#) & descrName = "Visual Cortex Eye Mvt", const [std::string](#) & tagName = "VisualCortexEyeMvt")**

Construct with no channel; channels are ignored anyways.

Definition at line 59 of file VisualCortexEyeMvt.C.

References [ModelComponent::addSubComponent\(\)](#).

**10.330.2.2** VisualCortexEyeMvt::VisualCortexEyeMvt ([ModelManager](#) & mgr, const [std::string](#) & descrName, const [std::string](#) & tagName, const double chanIw, const double chanCw, const double chanOw, const double chanFw, const double chanMw)

Construct, ignoring channel definitions.

Definition at line 80 of file VisualCortexEyeMvt.C.

References [ModelComponent::addSubComponent\(\)](#).

**10.330.2.3** VisualCortexEyeMvt::VisualCortexEyeMvt ([ModelManager](#) & mgr, const [std::string](#) & descrName, const [std::string](#) & tagName, const double chanIw, const double chanCw, const double chanCow, const double chanOw, const double chanFw, const double chanMw, const double chanLw, const double chanTw, const double chanXw, const double chanEw, const double chanDw)

Construct, ignoring even more channel definitions.

Definition at line 106 of file VisualCortexEyeMvt.C.

References [ModelComponent::addSubComponent\(\)](#).

**10.330.2.4** VisualCortexEyeMvt::~~VisualCortexEyeMvt () [virtual]

Virtual destructor for safe inheritance.

Definition at line 138 of file VisualCortexEyeMvt.C.

### 10.330.3 Member Function Documentation

**10.330.3.1** void VisualCortexEyeMvt::evolve (const double stime) [virtual]

evolve...

Let the major hacking begin! [VisualCortex](#) usually does not evolve, but here we need it to. [Brain](#) will call this evolve from its own to get us going.

Definition at line 241 of file VisualCortexEyeMvt.C.

References [ModelParam< bool >::getVal\(\)](#), [ModelParam< float >::getVal\(\)](#), [Point2D::i](#), [itsForgetFac](#), [itsSaccadeOnly](#), [Point2D::j](#), [takeMax\(\)](#), and [uint](#).

### 10.330.3.2 **void VisualCortexEyeMvt::exportOptions (const int *whatToExport*, const bool *recurse* = true) [virtual]**

Export some of our [ModelParam](#) parameters as command-line-options.

Reimplemented from [VisualCortex](#).

Definition at line 202 of file VisualCortexEyeMvt.C.

References [VisualCortex::exportOptions\(\)](#), [itsFnames](#), [itsForgetFac](#), [itsSaccadeOnly](#), [itsUseMax](#), [OPTEXP\\_CORE](#), and [ModelManager::requestOption\(\)](#).

### 10.330.3.3 **[Point2D](#) VisualCortexEyeMvt::getEyePos (const [uint](#) *human*) const**

Get current eye position of our of our humans.

Definition at line 321 of file VisualCortexEyeMvt.C.

References [ASSERT](#), and [uint](#).

### 10.330.3.4 **[Image](#)< float > VisualCortexEyeMvt::getOutput () [virtual]**

Get the output from visual cortex.

Reimplemented from [VisualCortex](#).

Definition at line 291 of file VisualCortexEyeMvt.C.

References [Image< T >::clear\(\)](#), [ModelParam< float >::getVal\(\)](#), [ModelParam< bool >::getVal\(\)](#), [itsUseMax](#), [takeMax\(\)](#), and [uint](#).

### 10.330.3.5 **void VisualCortexEyeMvt::input (const [Image](#)< [PixRGB](#)< [byte](#) > > & *inp*, const double *t*, const [Image](#)< [byte](#) > & *clipMask* = [Image](#)< [byte](#) >()) [virtual]**

Present new input (e.g., a new frame from a video stream):.

Reimplemented from [VisualCortex](#).

Definition at line 227 of file VisualCortexEyeMvt.C.

References [ImageSet< float >::size\(\)](#), and [uint](#).

### 10.330.3.6 **[uint](#) VisualCortexEyeMvt::numHumans () const**

Get our number of humans.

Definition at line 317 of file VisualCortexEyeMvt.C.

References [uint](#).

### 10.330.3.7 void VisualCortexEyeMvt::reset (const bool *recurse* = true) [virtual]

Reset [VisualCortex](#) and propagate the reset signal to the sub-components.

Reimplemented from [VisualCortex](#).

Definition at line 220 of file VisualCortexEyeMvt.C.

References [VisualCortex::reset\(\)](#).

### 10.330.3.8 void VisualCortexEyeMvt::start1 () [protected, virtual]

This is called from within [start\(\)](#) before the subcomponents start.

Reimplemented from [ModelComponent](#).

Definition at line 142 of file VisualCortexEyeMvt.C.

References [ModelComponent::addSubComponent\(\)](#), [SharedPtr< T >::dynCastFrom\(\)](#), [SharedPtr< T >::get\(\)](#), [ModelComponent::getModelParamVal\(\)](#), [ModelParam< int >::getVal\(\)](#), [itsFnames](#), [LevelSpec::mapLevel\(\)](#), [OPTEXP\\_ALL](#), [ImageSet< float >::reset\(\)](#), [SharedPtr< T >::reset\(\)](#), [ModelComponent::subComponent\(\)](#), and [uint](#).

## 10.330.4 Member Data Documentation

### 10.330.4.1 [ModelParam<int>](#) [VisualCortexEyeMvt::itsDelay](#) [protected]

human-to-human delay, in eye samples

Definition at line 111 of file VisualCortexEyeMvt.H.

### 10.330.4.2 [ModelParam<std::string>](#) [VisualCortexEyeMvt::itsFnames](#) [protected]

names of eye movement files

Definition at line 108 of file VisualCortexEyeMvt.H.

Referenced by [exportOptions\(\)](#), and [start1\(\)](#).

### 10.330.4.3 [ModelParam<float>](#) [VisualCortexEyeMvt::itsForgetFac](#) [protected]

forgetting factor

Definition at line 110 of file VisualCortexEyeMvt.H.

Referenced by `evolve()`, and `exportOptions()`.

**10.330.4.4**   [ModelParam<bool>](#)   [VisualCortexEyeMvt::itsSaccadeOnly](#)  
[protected]

plot only when a saccade

Definition at line 113 of file `VisualCortexEyeMvt.H`.

Referenced by `evolve()`, and `exportOptions()`.

**10.330.4.5**   [ModelParam<float>](#)   [VisualCortexEyeMvt::itsSigma](#)  
[protected]

sigma of gaussian blobs in pix at sml

Definition at line 109 of file `VisualCortexEyeMvt.H`.

**10.330.4.6**   [ModelParam<bool>](#)   [VisualCortexEyeMvt::itsUseMax](#)  
[protected]

use max to combine instead of sum

Definition at line 112 of file `VisualCortexEyeMvt.H`.

Referenced by `exportOptions()`, and `getOutput()`.

The documentation for this class was generated from the following files:

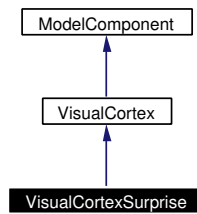
- [VisualCortexEyeMvt.H](#)
- [VisualCortexEyeMvt.C](#)

## 10.331 VisualCortexSurprise Class Reference

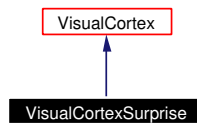
This is a derivation of [VisualCortex](#) using [SingleChannelSurprise](#) channels.

```
#include <VisualCortexSurprise.H>
```

Inheritance diagram for VisualCortexSurprise:



Collaboration diagram for VisualCortexSurprise:



### Public Member Functions

- [VisualCortexSurprise](#) ([ModelManager](#) &mgr, const [std::string](#) &descrName, const [std::string](#) &tagName)  
*Constructor without any channels.*
- [VisualCortexSurprise](#) ([ModelManager](#) &mgr, const [std::string](#) &descrName, const [std::string](#) &tagName, const double chanIw, const double chanCw, const double chanOw, const double chanFw, const double chanMw)  
*Construct with default intens/color/orient/flicker/motion surprise chans.*
- [VisualCortexSurprise](#) ([ModelManager](#) &mgr, const [std::string](#) &descrName, const [std::string](#) &tagName, const double chanIw, const double chanCw, const double chanCow, const double chanOw, const double chanFw, const double chanMw, const double chanLw, const double chanTw, const double chanXw, const double chanEw, const double chanDw)  
*Construct with default Surprise channels.*

- virtual [~VisualCortexSurprise](#) ()

*Virtual destructor for safe inheritance.*

- virtual void [addDefaultChannels](#) (const double chanIw=1.0, const double chanCw=1.0, const double chanCow=1.0, const double chanOw=1.0, const double chanFw=1.0, const double chanMw=1.0, const double chanLw=0.0, const double chanTw=0.0, const double chanXw=0.0, const double chanEw=0.0, const double chanDw=0.0)

*Adds the default channels to the VisualCortexSurprise object.*

### 10.331.1 Detailed Description

This is a derivation of [VisualCortex](#) using [SingleChannelSurprise](#) channels.

It computes Surprise instead of computing saliency.

Definition at line 47 of file VisualCortexSurprise.H.

### 10.331.2 Constructor & Destructor Documentation

#### 10.331.2.1 [VisualCortexSurprise::VisualCortexSurprise](#) ([ModelManager](#) & *mgr*, const [std::string](#) & *descrName*, const [std::string](#) & *tagName*)

Constructor without any channels.

Definition at line 43 of file VisualCortexSurprise.C.

#### 10.331.2.2 [VisualCortexSurprise::VisualCortexSurprise](#) ([ModelManager](#) & *mgr*, const [std::string](#) & *descrName*, const [std::string](#) & *tagName*, const double *chanIw*, const double *chanCw*, const double *chanOw*, const double *chanFw*, const double *chanMw*)

Construct with default intens/color/orient/flicker/motion surprise chans.

Definition at line 51 of file VisualCortexSurprise.C.

References [addDefaultChannels\(\)](#).



**10.331.2.3** **VisualCortexSurprise::VisualCortexSurprise** ([ModelManager](#) & mgr, const [std::string](#) & descrName, const [std::string](#) & tagName, const double chanIw, const double chanCw, const double chanCow, const double chanOw, const double chanFw, const double chanMw, const double chanLw, const double chanTw, const double chanXw, const double chanEw, const double chanDw)

Construct with default Surprise channels.

Definition at line 64 of file VisualCortexSurprise.C.

References [addDefaultChannels\(\)](#).

**10.331.2.4** **VisualCortexSurprise::~~VisualCortexSurprise** () [virtual]

Virtual destructor for safe inheritance.

Definition at line 78 of file VisualCortexSurprise.C.

### 10.331.3 Member Function Documentation

**10.331.3.1** **void VisualCortexSurprise::addDefaultChannels** (const double chanIw = 1.0, const double chanCw = 1.0, const double chanCow = 1.0, const double chanOw = 1.0, const double chanFw = 1.0, const double chanMw = 1.0, const double chanLw = 0.0, const double chanTw = 0.0, const double chanXw = 0.0, const double chanEw = 0.0, const double chanDw = 0.0) [virtual]

Adds the default channels to the VisualCortexSurprise object.

Definition at line 82 of file VisualCortexSurprise.C.

References [VisualCortex::addChannel\(\)](#), [VisualCortex::channel\(\)](#), [makeShared\(\)](#), and [ChannelBase::setTotalWeight\(\)](#).

Referenced by [VisualCortexSurprise\(\)](#).

The documentation for this class was generated from the following files:

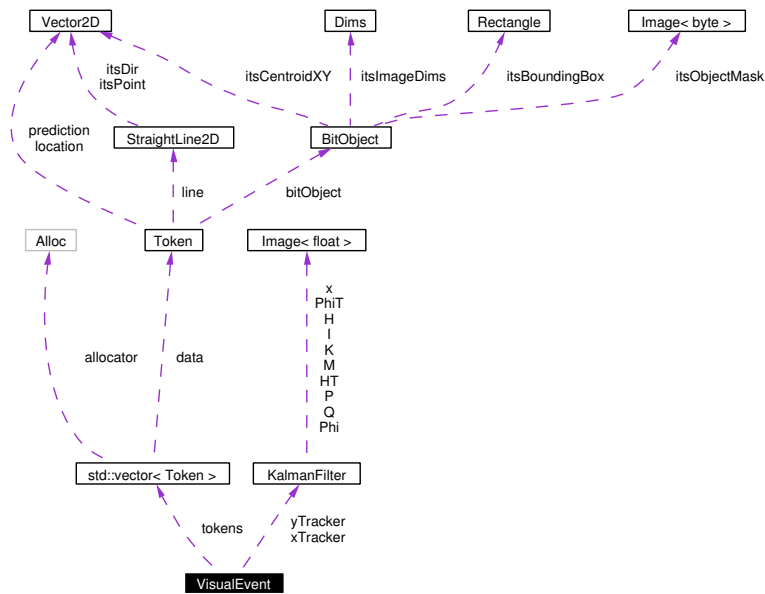
- [VisualCortexSurprise.H](#)
- [VisualCortexSurprise.C](#)

## 10.332 VisualEvent Class Reference

stores a visual event as a sequence of tokens with additional information

```
#include <VisualEvent.H>
```

Collaboration diagram for VisualEvent:



### Public Member Functions

- **VisualEvent** (**Token** tk, int maxDist)  
*constructor*
- **VisualEvent** (std::istream &is)  
*read the VisualEvent from the input stream is*
- void **writeToStream** (std::ostream &os) const  
*write the entire VisualEvent to the output stream os*
- void **readFromStream** (std::istream &is)  
*read the VisualEvent from the input stream is*
- void **writePositions** (std::ostream &os) const

*write all the positions for this event to the output stream os*

- [Point2D predictedLocation](#) () const  
*get the prediction for the location of the next token*
- bool [isTokenOk](#) (const [Token](#) &tk) const  
*is this token allowed as the next one?*
- float [getCost](#) (const [Token](#) &tk) const  
*returns the cost of associating tk with this event*
- void [assign](#) (const [Token](#) &tk, const [Vector2D](#) &foe)  
*assign tk to this event, use foe as the focus of expansion*
- bool [doesIntersect](#) (const [BitObject](#) &obj, int frameNum) const  
*if the [BitObject](#) intersects with the one for this event at frameNum*
- void [close](#) ()  
*mark this event as "closed"*
- bool [isClosed](#) ()  
*whether this event is closed*
- uint [getEventNum](#) () const  
*return the event identification number of this event*
- uint [getStartFrame](#) () const  
*return the frame number of the first token*
- uint [getEndFrame](#) () const  
*return the frame number of the last token*
- uint [getNumberOfFrames](#) () const  
*return the number of frames that this event spans*
- int [getMaxSize](#) () const  
*return the maximum object size of any of the tokens*
- [Token getMaxSizeToken](#) () const  
*return the token that has the maximum object size*
- [Token getToken](#) (const uint frame\_num) const

*return a token based on a frame number*

- bool `isFrameOk` (const uint frame\_num) const  
*return whether frame\_num is between start frame and end frame*
- `std::vector< float > getPropertyVector` ()  
*return a vector that contains a bunch of numbers that describe the event*
- `Dims getMaxObjectDims` () const  
*returns the maximum dimensions of the tracked object in any of the frames*

### 10.332.1 Detailed Description

stores a visual event as a sequence of tokens with additional information  
Definition at line 146 of file VisualEvent.H.

### 10.332.2 Constructor & Destructor Documentation

#### 10.332.2.1 VisualEvent::VisualEvent (Token tk, int maxDist)

constructor

##### Parameters:

**tk** the first token for this event

Definition at line 173 of file VisualEvent.C.

References `Token::bitObject`, `BitObject::getArea()`, `Token::location`, `Vector2D::x()`, and `Vector2D::y()`.

#### 10.332.2.2 VisualEvent::VisualEvent (std::istream & is)

read the VisualEvent from the input stream is

Definition at line 193 of file VisualEvent.C.

References `readFromStream()`.

### 10.332.3 Member Function Documentation

#### 10.332.3.1 void VisualEvent::assign (const Token & tk, const Vector2D & foe)

assign tk to this event, use foe as the focus of expansion

Definition at line 282 of file VisualEvent.C.

References ASSERT, Token::bitObject, Token::frame\_nr, BitObject::getArea(), KalmanFilter::getEstimate(), isTokenOk(), Vector2D::isValid(), Token::location, KalmanFilter::update(), Vector2D::x(), and Vector2D::y().

#### 10.332.3.2 void VisualEvent::close () [inline]

mark this event as "closed"

Definition at line 360 of file VisualEvent.H.

#### 10.332.3.3 bool VisualEvent::doesIntersect (const BitObject & obj, int frameNum) const

if the BitObject intersects with the one for this event at frameNum

Definition at line 312 of file VisualEvent.C.

References Token::bitObject, BitObject::doesIntersect(), getToken(), and isFrameOk().

#### 10.332.3.4 float VisualEvent::getCost (const Token & tk) const

returns the cost of associating tk with this event

##### Returns:

returns -1.0F if the token is not valid for this event

Definition at line 268 of file VisualEvent.C.

References KalmanFilter::getCost(), KalmanFilter::getEstimate(), isTokenOk(), Token::location, Vector2D::x(), and Vector2D::y().

#### 10.332.3.5 uint VisualEvent::getEndFrame () const [inline]

return the frame number of the last token

Definition at line 376 of file VisualEvent.H.

References uint.

#### 10.332.3.6 uint VisualEvent::getEventNum () const [inline]

return the event identification number of this event

Definition at line 368 of file VisualEvent.H.

References uint.

Referenced by getPropertyVector(), and MbariResultViewer::saveSingleEventFrame().

#### 10.332.3.7 **Dims** VisualEvent::getMaxObjectDims () const

returns the maximum dimensions of the tracked object in any of the frames

Definition at line 385 of file VisualEvent.C.

References Dims::h(), and Dims::w().

Referenced by MbariResultViewer::saveSingleEventFrame().

#### 10.332.3.8 **int** VisualEvent::getMaxSize () const [inline]

return the maximum object size of any of the tokens

Definition at line 384 of file VisualEvent.H.

#### 10.332.3.9 **Token** VisualEvent::getMaxSizeToken () const [inline]

return the token that has the maximum object size

Definition at line 388 of file VisualEvent.H.

References getToken().

Referenced by getPropertyVector().

#### 10.332.3.10 **uint** VisualEvent::getNumberOfFrames () const [inline]

return the number of frames that this event spans

Definition at line 380 of file VisualEvent.H.

References uint.

#### 10.332.3.11 **std::vector< float >** VisualEvent::getPropertyVector ()

return a vector that contains a bunch of numbers that describe the event

see source code for details

Definition at line 320 of file VisualEvent.C.

References Token::angle, Token::bitObject, BitObject::getArea(), BitObject::getElongation(), getEventNum(), BitObject::getMajorAxis(), BitObject::getMaxMinAvgIntensity(), getMaxSizeToken(), BitObject::getMinorAxis(), BitObject::getOrientation(), BitObject::getSecondMoments(), BitObject::isValid(), and uint.

#### 10.332.3.12 **uint** VisualEvent::getStartFrame () const [inline]

return the frame number of the first token

Definition at line 372 of file VisualEvent.H.

References uint.

#### 10.332.3.13 **Token** VisualEvent::getToken (const **uint** frame\_num) const [inline]

return a token based on a frame number

Definition at line 392 of file VisualEvent.H.

References ASSERT, isFrameOk(), and uint.

Referenced by doesIntersect(), getMaxSizeToken(), and MbariResultViewer::saveSingleEventFrame().

#### 10.332.3.14 **bool** VisualEvent::isClosed () [inline]

whether this event is closed

Definition at line 364 of file VisualEvent.H.

#### 10.332.3.15 **bool** VisualEvent::isFrameOk (const **uint** frame\_num) const [inline]

return whether frame\_num is between start frame and end frame

Definition at line 399 of file VisualEvent.H.

References uint.

Referenced by doesIntersect(), getToken(), and MbariResultViewer::saveSingleEventFrame().

#### 10.332.3.16 **bool** VisualEvent::isTokenOk (const **Token** & tk) const

is this token allowed as the next one?

Definition at line 262 of file VisualEvent.C.

References Token::frame\_nr.

Referenced by assign(), and getCost().

#### **10.332.3.17 [Point2D](#) VisualEvent::predictedLocation () const**

get the prediction for the location of the next token

Definition at line 254 of file VisualEvent.C.

References KalmanFilter::getEstimate().

#### **10.332.3.18 void VisualEvent::readFromStream (std::istream & is)**

read the VisualEvent from the input stream is

Definition at line 221 of file VisualEvent.C.

References KalmanFilter::readFromStream().

Referenced by VisualEvent().

#### **10.332.3.19 void VisualEvent::writePositions (std::ostream & os) const**

write all the positions for this event to the output stream os

Definition at line 245 of file VisualEvent.C.

References uint.

#### **10.332.3.20 void VisualEvent::writeToStream (std::ostream & os) const**

write the entire VisualEvent to the output stream os

Definition at line 199 of file VisualEvent.C.

References uint, and KalmanFilter::writeToStream().

The documentation for this class was generated from the following files:

- [VisualEvent.H](#)
- [VisualEvent.C](#)



## 10.333 VisualEventSet Class Reference

contains a set of visual events to which a set of tokens can be matched

```
#include <VisualEvent.H>
```

Collaboration diagram for VisualEventSet:

### Public Member Functions

- [VisualEventSet](#) (const int maxDist, const [uint](#) minFrameNum, const int minSize, const [std::string](#) &filename)  
*constructor*
- [VisualEventSet](#) (std::istream &is)  
*read the VisualEventSet from the input stream is*
- void [writeToStream](#) (std::ostream &os) const  
*write the entire VisualEventSet to the output stream os*
- void [readFromStream](#) (std::istream &is)  
*read the VisualEventSet from the input stream is*
- void [writePositions](#) (std::ostream &os) const  
*write th positions of all events to the output stream os*
- void [updateEvents](#) (const [Image](#)< [byte](#) > &binMap, const [Vector2D](#) &curFOE, int frameNum)  
*extract BitObjects from binMap and assign them to the events*
- void [initiateEvents](#) ([std::list](#)< [BitObject](#) > &bos, int frameNum)  
*initiate new events for all BitObjects in bos if they aren't tracked yet*
- bool [doesIntersect](#) (const [BitObject](#) &obj, int frameNum) const  
*if obj intersects with any of the event at frameNum*
- [Vector2D](#) [getFOE](#) (int frameNum) const  
*returns the current focus of expansion for frameNum*
- [uint](#) [numEvents](#) () const  
*return the number of stored events*
- void [reset](#) ()

*delete all stored events*

- void `cleanUp` (uint currFrame, uint maxFrameSkip=1)  
*clean up the event list - erase all unsuccessful candidates*
- void `closeAll` ()  
*close all events (for clean-up at the end)*
- `std::vector< Token > getTokens` (uint frameNum)  
*returns a set of all tokens stored at a particular frame number*
- void `drawTokens` (Image< PixRGB< byte > > &img, uint frameNum, PropertyVectorSet &pvs, int circleRadius, BitObjectDrawMode mode, float opacity, PixRGB< byte > colorInteresting, PixRGB< byte > colorCandidate, PixRGB< byte > colorPred, PixRGB< byte > colorFOE, bool showEvent-Labels)  
*draw the tokens for a particular frame number into an image*
- PropertyVectorSet `getPropertyVectorSet` ()  
*returns a PropertyVectorSet for this VisualEventSet*
- int `getAllClosedFrameNum` (uint currFrame)  
*return the latest frame number before currFrame for which events have been closed already*
- bool `isEventInteresting` (std::vector< float > propVec) const  
*whether event is "interesting"*
- bool `doesEventExist` (uint eventNum) const  
*test whether the event with eventNum exists in this set*
- VisualEvent `getEventByNumber` (uint eventNum) const  
*Returns the event with number eventNum.*

## Protected Member Functions

- `std::vector< std::list< VisualEvent >::iterator > getEventsForFrame` (uint framenum)  
*Returns a vector with iterators pointing to all events at framenum.*

### 10.333.1 Detailed Description

contains a set of visual events to which a set of tokens can be matched

Definition at line 235 of file VisualEvent.H.

### 10.333.2 Constructor & Destructor Documentation

#### 10.333.2.1 VisualEventSet::VisualEventSet (const int *maxDist*, const uint *minFrameNum*, const int *minSize*, const std::string & *filename*)

constructor

##### Parameters:

***maxDist*** the maximum distance that a token can have from an event and still be considered for becoming part of the event

***minFrameNum*** the minimum number of frames for which an event needs be present

***minSize*** the minimum size (in pixels) for an "interesting" event

***fileName*** the fileName for saving intermediate result

Definition at line 401 of file VisualEvent.C.

References uint.

#### 10.333.2.2 VisualEventSet::VisualEventSet (std::istream & *is*)

read the VisualEventSet from the input stream is

Definition at line 417 of file VisualEvent.C.

References readFromStream().

### 10.333.3 Member Function Documentation

#### 10.333.3.1 void VisualEventSet::cleanUp (uint *currFrame*, uint *maxFrameSkip* = 1)

clean up the event list - erase all unsuccessful candidates

##### Parameters:

***currFrame*** - the current frame number in processing

***maxFrameSkip*** how many frames may be skipped using interpolation (default: 1)

Definition at line 723 of file VisualEvent.C.

References uint.

### 10.333.3.2 void VisualEventSet::closeAll ()

close all events (for clean-up at the end)

Definition at line 741 of file VisualEvent.C.

### 10.333.3.3 bool VisualEventSet::doesEventExist (uint eventNum) const

test whether the event with eventNum exists in this set

Definition at line 941 of file VisualEvent.C.

References uint.

### 10.333.3.4 bool VisualEventSet::doesIntersect (const BitObject & obj, int frameNum) const

if obj intersects with any of the event at frameNum

Definition at line 594 of file VisualEvent.C.

Referenced by initiateEvents(), and updateEvents().

### 10.333.3.5 void VisualEventSet::drawTokens (Image< PixRGB< byte > > & img, uint frameNum, PropertyVectorSet & pvs, int circleRadius, BitObjectDrawMode mode, float opacity, PixRGB< byte > colorInteresting, PixRGB< byte > colorCandidate, PixRGB< byte > colorPred, PixRGB< byte > colorFOE, bool showEventLabels)

draw the tokens for a particular frame number into an image

#### Parameters:

*img* the image to draw on

*frameNum* the frame number for which to draw the tokens

*circleRadius* the radius of the circle used to mark the tokens

*colorInteresting* the color for "interesting" tokens

*colorCandidate* the color for unsuccessful candidate tokens

*colorPred* the color for event location predictions

*showEventLabels* whether to write the event labels into the image

Definition at line 765 of file VisualEvent.C.

References `Token::bitObject`, `Image< T >::clear()`, `BitObject::draw()`, `BitObject::getArea()`, `BitObject::getBoundingBox()`, `Image< T >::getDims()`, `getFOE()`, `Vector2D::getPoint2D()`, `PropertyVectorSet::getPropertyVectorForEvent()`, `Point2D::i`, `isEventInteresting()`, `Vector2D::isValid()`, `BitObject::isValid()`, `Point2D::j`, `Token::location`, `Rectangle::modify()`, `NO_INIT`, `pasteImage()`, `Token::prediction`, `replaceVals()`, `Image< T >::resize()`, `toStr()`, `uint`, and `Image< T >::writeText()`.

Referenced by `MbariResultViewer::outputResultFrame()`.

#### 10.333.3.6 `int VisualEventSet::getAllClosedFrameNum (uint currFrame)`

return the latest frame number before currFrame for which events have been closed already

Definition at line 906 of file VisualEvent.C.

References `uint`.

#### 10.333.3.7 `VisualEvent VisualEventSet::getEventByNumber (uint eventNum) const`

Returns the event with number eventNum.

Definition at line 951 of file VisualEvent.C.

References `uint`.

#### 10.333.3.8 `std::vector< std::list< VisualEvent >::iterator > VisualEventSet::getEventsForFrame (uint framenum) [protected]`

Returns a vector with iterators pointing to all events at framenum.

Definition at line 964 of file VisualEvent.C.

References `uint`.

#### 10.333.3.9 `Vector2D VisualEventSet::getFOE (int frameNum) const`

returns the current focus of expansion for frameNum

Definition at line 699 of file VisualEvent.C.

References `ASSERT`.

Referenced by `drawTokens()`.

**10.333.3.10   [PropertyVectorSet](#) VisualEventSet::getPropertyVectorSet ()**

returns a [PropertyVectorSet](#) for this VisualEventSet

Definition at line 892 of file VisualEvent.C.

References [PropertyVectorSet::itsVectors](#).

**10.333.3.11   [std::vector](#)< [Token](#) > VisualEventSet::getTokens ([uint](#) *frameNum*)**

returns a set of all tokens stored at a particular frame number

Definition at line 749 of file VisualEvent.C.

References [uint](#).

**10.333.3.12   void VisualEventSet::initiateEvents ([std::list](#)< [BitObject](#) > & *bos*,  
int *frameNum*)**

initiate new events for all BitObjects in bos if they aren't tracked yet

Definition at line 555 of file VisualEvent.C.

References [ASSERT](#), and [doesIntersect\(\)](#).

**10.333.3.13   bool VisualEventSet::isEventInteresting ([std::vector](#)< float >  
*propVec*) const**

whether event is "interesting"

Definition at line 927 of file VisualEvent.C.

Referenced by [drawTokens\(\)](#).

**10.333.3.14   [uint](#) VisualEventSet::numEvents () const**

return the number of stored events

Definition at line 711 of file VisualEvent.C.

References [uint](#).

**10.333.3.15   void VisualEventSet::readFromStream ([std::istream](#) & *is*)**

read the VisualEventSet from the input stream is

Definition at line 445 of file VisualEvent.C.

Referenced by `MbariResultViewer::loadVisualEventSet()`, and `VisualEventSet()`.

#### 10.333.3.16 `void VisualEventSet::reset ()`

delete all stored events

Definition at line 717 of file `VisualEvent.C`.

#### 10.333.3.17 `void VisualEventSet::updateEvents (const Image< byte > & binMap, const Vector2D & curFOE, int frameNum)`

extract BitObjects from binMap and assign them to the events

##### Parameters:

*curFOE* the current focus of expansion for detecting unusual motion

Definition at line 475 of file `VisualEvent.C`.

References `ASSERT`, `doesIntersect()`, `extractBitObjects()`, `Image< T >::getDims()`, `Image< T >::getHeight()`, `Image< T >::getWidth()`, `Point2D::i`, and `Point2D::j`.

#### 10.333.3.18 `void VisualEventSet::writePositions (std::ostream & os) const`

write th positions of all events to the output stream os

Definition at line 467 of file `VisualEvent.C`.

Referenced by `MbariResultViewer::savePositions()`.

#### 10.333.3.19 `void VisualEventSet::writeToStream (std::ostream & os) const`

write the entire VisualEventSet to the output stream os

Definition at line 423 of file `VisualEvent.C`.

References `uint`.

Referenced by `MbariResultViewer::saveVisualEventSet()`.

The documentation for this class was generated from the following files:

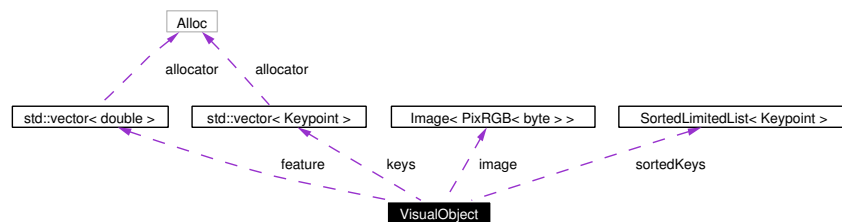
- [VisualEvent.H](#)
- [VisualEvent.C](#)

## 10.334 VisualObject Class Reference

describes name and attributes of a visual object

```
#include <VisualObject.H>
```

Collaboration diagram for VisualObject:



### Public Member Functions

- [VisualObject](#) (int sortOrder=0)  
*Constructor.*
- [VisualObject](#) (const char \*n, int sortOrder=0)  
*Constructor.*
- [VisualObject](#) (FILE \*fp, int sortOrder=0)  
*Construct by reading file.*
- [VisualObject](#) (const [VisualObject](#) &vo)  
*Copy constructor.*
- [~VisualObject](#) ()  
*Destructor.*
- [VisualObject](#) & [operator=](#) (const [VisualObject](#) &vo)  
*Assignment constructor.*
- const char \* [getName](#) () const  
*name getter*
- void [setName](#) (char \*n)  
*name setter*



- const char \* [getImageFilename](#) () const  
*image filename getter*
- void [setImageFilename](#) (char \*imgFilename)  
*image filename setter*
- void [save](#) (FILE \*fp) const  
*save object to file*
- const [Image](#)< [PixRGB](#)< byte > > & [getImage](#) () const  
*image getter*
- void [setImage](#) ([Image](#)< [PixRGB](#)< byte > > &img)  
*image setter*
- void [setFeature](#) (const [std::vector](#)< double > &f)  
*features setter*
- double [getFeatureDistSq](#) (const [VisualObject](#) &obj) const  
*get the distance squared between the closest two feature vectors from*
- const [std::vector](#)< double > & [getFeature](#) () const  
*get feature (needed to add features to matched object)*
- const [VisualObjectMatch](#) [match](#) ([VisualObject](#) &obj, char \*matchingAlg, int recogMinMatch, bool sortKeys)  
*find match using the given object and algorithm*
- void [match](#) (const [std::vector](#)< Keypoint > &objKeys, [VisualObjectMatch](#) &objMatch, char \*matchingAlg, int recogMinMatch, bool sortKeys)  
*find match using the given keypoints and algorithm*
- int [createKeys](#) (float x, float y, float s, int dimX, int dimY, float octSize, double magnitude, [Image](#)< float > &grad, [Image](#)< float > &orie)  
*Creates object keys. Returns the number created.*
- void [incTimesRecognized](#) ()  
*Increments the number of times this object has been recognized.*
- const int [getTimesRecognized](#) () const  
*Returns the number of times this object has been recognized.*

### 10.334.1 Detailed Description

describes name and attributes of a visual object

Definition at line 131 of file VisualObject.H.

### 10.334.2 Constructor & Destructor Documentation

#### 10.334.2.1 VisualObject::VisualObject (int *sortOrder* = 0)

Constructor.

Definition at line 238 of file VisualObject.C.

#### 10.334.2.2 VisualObject::VisualObject (const char \* *n*, int *sortOrder* = 0)

Constructor.

Definition at line 247 of file VisualObject.C.

References setName().

#### 10.334.2.3 VisualObject::VisualObject (FILE \* *fp*, int *sortOrder* = 0)

Construct by reading file.

Definition at line 257 of file VisualObject.C.

References SortedLimitedList< Keypoint >::addUnsorted(), SortedLimitedList< Keypoint >::getSize(), Raster::ReadRGB(), setImageFilename(), setName(), and SortedLimitedList< Keypoint >::sort().

#### 10.334.2.4 VisualObject::VisualObject (const [VisualObject](#) & vo)

Copy constructor.

Definition at line 313 of file VisualObject.C.

References feature, image, imageFilename, Image< PixRGB< byte > >::initialized(), keys, name, setImageFilename(), setName(), sortedKeys, and timesRecognized.

#### 10.334.2.5 VisualObject::~[VisualObject](#) ()

Destructor.

Definition at line 328 of file VisualObject.C.

### 10.334.3 Member Function Documentation

**10.334.3.1** `int VisualObject::createKeys (float x, float y, float s, int dimX, int dimY, float octSize, double magnitude, Image< float > & grad, Image< float > & orie)`

Creates object keys. Returns the number created.

Definition at line 520 of file VisualObject.C.

References SortedLimitedList< Keypoint >::addUnsorted(), Histogram::addValue(), Image< PixRGB< byte > >::drawCircle(), exp(), Histogram::findMax(), SortedLimitedList< Keypoint >::getSize(), Image< T >::getVal(), Histogram::getValue(), Histogram::smooth(), and SortedLimitedList< Keypoint >::sort().

**10.334.3.2** `const std::vector< double > & VisualObject::getFeature () const`

get feature (needed to add features to matched object)

Definition at line 414 of file VisualObject.C.

Referenced by getFeatureDistSq().

**10.334.3.3** `double VisualObject::getFeatureDistSq (const VisualObject & obj) const`

get the distance squared between the closest two feature vectors from this object and the given object

Definition at line 397 of file VisualObject.C.

References getFeature(), and uint.

**10.334.3.4** `const Image< PixRGB< byte > > & VisualObject::getImage () const`

image getter

Definition at line 377 of file VisualObject.C.

Referenced by match().

**10.334.3.5** `const char * VisualObject::getImageFilename () const`

image filename getter

Definition at line 363 of file VisualObject.C.

**10.334.3.6   const char \* VisualObject::getName () const**

name getter

Definition at line 349 of file VisualObject.C.

Referenced by VisualObjectDB::addObject(), KnowledgeBase::addVisualObject(), and BPnnnet::recognize().

**10.334.3.7   const int VisualObject::getTimesRecognized () const**

Returns the number of times this object has been recognized.

Definition at line 426 of file VisualObject.C.

**10.334.3.8   void VisualObject::incTimesRecognized ()**

Increments the number of times this object has been recognized.

Definition at line 420 of file VisualObject.C.

**10.334.3.9   void VisualObject::match (const [std::vector](#)< Keypoint > & objKeys, [VisualObjectMatch](#) & objMatch, char \* matchingAlg, int recogMinMatch, bool sortKeys)**

find match using the given keypoints and algorithm

Definition at line 486 of file VisualObject.C.

References [KDTree::createKDTree\(\)](#), [VisualObjectMatch::getMatchCount\(\)](#), [VisualObjectMatch::incrementNumberKeyCompare\(\)](#), [VisualObjectMatch::match\(\)](#), [SortedList< Keypoint >::toVector\(\)](#), and [uint](#).

**10.334.3.10   const [VisualObjectMatch](#) VisualObject::match ([VisualObject](#) & obj, char \* matchingAlg, int recogMinMatch, bool sortKeys)**

find match using the given object and algorithm

Definition at line 457 of file VisualObject.C.

References [VisualObjectMatch::calculateScore\(\)](#), [combineImagesVertically\(\)](#), [getImage\(\)](#), [Image< PixRGB< byte > >::initialized\(\)](#), [VisualObjectMatch::setObjectName\(\)](#), [VisualObjectMatch::setOutput\(\)](#), [VisualObjectMatch::setOutputVerticalOffset\(\)](#), and [SortedList< Keypoint >::toVector\(\)](#).

**10.334.3.11 VisualObject & VisualObject::operator= (const VisualObject & vo)**

Assignment constructor.

Definition at line 333 of file VisualObject.C.

References feature, image, imageFilename, Image< PixRGB< byte > >::initialized(), keys, name, setImageFilename(), setName(), sortedKeys, and timesRecognized.

**10.334.3.12 void VisualObject::save (FILE \* fp) const**

save object to file

Definition at line 432 of file VisualObject.C.

References Image< PixRGB< byte > >::initialized(), uint, and Raster::WriteRGB().

**10.334.3.13 void VisualObject::setFeature (const std::vector< double > & f)**

features setter

Definition at line 389 of file VisualObject.C.

Referenced by InferenceTemporal::input().

**10.334.3.14 void VisualObject::setImage (Image< PixRGB< byte > > & img)**

image setter

Definition at line 383 of file VisualObject.C.

Referenced by InferenceTemporal::input().

**10.334.3.15 void VisualObject::setImageFilename (char \* imgFilename)**

image filename setter

Definition at line 369 of file VisualObject.C.

Referenced by operator=(), VisualObjectDB::train(), and VisualObject().

**10.334.3.16 void VisualObject::setName (char \* n)**

name setter

Definition at line 355 of file VisualObject.C.

Referenced by operator=(), VisualObjectDB::train(), and VisualObject().

The documentation for this class was generated from the following files:

- [VisualObject.H](#)
- [VisualObject.C](#)

## 10.335 VisualObjectDB Class Reference

VisualObjectDatabase.

```
#include <VisualObjectDB.H>
```

### Public Member Functions

- [VisualObjectDB](#) ()  
*Constructor.*
- [VisualObjectDB](#) (const char \*filename, int sortOrder=0)  
*Construct from file.*
- [~VisualObjectDB](#) ()  
*Destructor.*
- const [VisualObject](#) & [getObject](#) (int index) const  
*object getter*
- const [VisualObject](#) & [getObject](#) (char \*objectName) const  
*object getter*
- bool [addObject](#) ([VisualObject](#) &obj)  
*add object*
- int [getSize](#) () const  
*get number of objects in the database*
- void [save](#) (const char \*filename) const  
*save to file*
- const [VisualObjectMatch](#) [matchObject](#) ([VisualObject](#) &obj, int recogMinMatch, char \*matchingAlg, bool sortKeys, bool sortObjects)  
*find matches for the keypoints in the scene*
- void [train](#) ([VisualObject](#) &obj)  
*train the database on a new object*
- void [filter](#) (int recogThresh)  
*filter the database based on the number of times an object has been*

### 10.335.1 Detailed Description

VisualObjectDatabase.

Definition at line 46 of file VisualObjectDB.H.

### 10.335.2 Constructor & Destructor Documentation

#### 10.335.2.1 VisualObjectDB::VisualObjectDB ()

Constructor.

Definition at line 47 of file VisualObjectDB.C.

#### 10.335.2.2 VisualObjectDB::VisualObjectDB (const char \* *filename*, int *sortOrder* = 0)

Construct from file.

Definition at line 51 of file VisualObjectDB.C.

#### 10.335.2.3 VisualObjectDB::~VisualObjectDB ()

Destructor.

Definition at line 73 of file VisualObjectDB.C.

### 10.335.3 Member Function Documentation

#### 10.335.3.1 bool VisualObjectDB::addObject (VisualObject & *obj*)

add object

Definition at line 100 of file VisualObjectDB.C.

References VisualObject::getName(), and uint.

Referenced by train().

#### 10.335.3.2 void VisualObjectDB::filter (int *recogThresh*)

filter the database based on the number of times an object has been successfully recognized

Definition at line 200 of file VisualObjectDB.C.



References uint.

#### 10.335.3.3 **const VisualObject & VisualObjectDB::getObject (char \* *objectName*) const**

object getter

Definition at line 85 of file VisualObjectDB.C.

References uint.

#### 10.335.3.4 **const VisualObject & VisualObjectDB::getObject (int *index*) const**

object getter

Definition at line 79 of file VisualObjectDB.C.

#### 10.335.3.5 **int VisualObjectDB::getSize () const**

get number of objects in the database

Definition at line 112 of file VisualObjectDB.C.

#### 10.335.3.6 **const VisualObjectMatch VisualObjectDB::matchObject (VisualObject & *obj*, int *recogMinMatch*, char \* *matchingAlg*, bool *sortKeys*, bool *sortObjects*)**

find matche for the keypoints in the scene

Definition at line 135 of file VisualObjectDB.C.

References VisualObjectMatch::getMatchCount(), VisualObjectMatch::getNumberKeyCompare(), VisualObjectMatch::incrementNumberKeyCompare(), and uint.

Referenced by IneroTemporal::input().

#### 10.335.3.7 **void VisualObjectDB::save (const char \* *filename*) const**

save to file

Definition at line 118 of file VisualObjectDB.C.

References uint.

Referenced by IneroTemporal::input().

**10.335.3.8 void VisualObjectDB::train ([VisualObject](#) & *obj*)**

train the database on a new object

Definition at line 188 of file VisualObjectDB.C.

References [addObject\(\)](#), [VisualObject::setImageFilename\(\)](#), and [VisualObject::setName\(\)](#).

Referenced by [InferoTemporal::input\(\)](#).

The documentation for this class was generated from the following files:

- [VisualObjectDB.H](#)
- [VisualObjectDB.C](#)

## 10.336 VisualObjectMatch Class Reference

VisualObjectMatch.

```
#include <VisualObject.H>
```

Collaboration diagram for VisualObjectMatch:

### Public Member Functions

- [VisualObjectMatch](#) ()  
*constructor*
- const char \* [getObjectName](#) () const  
*get object name*
- void [setObjectName](#) (char \*name)  
*set object name*
- const [Image](#)< [PixRGB](#)< byte > > & [getOutput](#) () const  
*get output*
- void [setOutput](#) ([Image](#)< [PixRGB](#)< byte > > &img)  
*set output*
- const int [getMatchCount](#) () const  
*get count*
- const float [getScore](#) () const  
*get score*
- void [calculateScore](#) ()  
*set score*
- void [match](#) (const Keypoint &k1, const Keypoint &k2)  
*update output with a match*
- void [writeOutput](#) (char \*outputFile)  
*write output*
- void [setOutputVerticalOffset](#) (int value)  
*Set the vertical offset of the output (height of the top image).*

- void [incrementNumberKeyCompare](#) (int incr)  
*Increment the number of keypoint comparisons made.*
- int [getNumberKeyCompare](#) () const  
*Get the number of keypoint comparisons made.*

### 10.336.1 Detailed Description

VisualObjectMatch.

Definition at line 79 of file VisualObject.H.

### 10.336.2 Constructor & Destructor Documentation

#### 10.336.2.1 VisualObjectMatch::VisualObjectMatch ()

constructor

Definition at line 113 of file VisualObject.C.

### 10.336.3 Member Function Documentation

#### 10.336.3.1 void VisualObjectMatch::calculateScore ()

set score

Definition at line 170 of file VisualObject.C.

References [KeypointMatch::getDistanceSq\(\)](#), [SortedLimitedList< KeypointMatch >::getSize\(\)](#), and [SortedLimitedList< KeypointMatch >::getValue\(\)](#).

Referenced by [VisualObject::match\(\)](#), and [writeOutput\(\)](#).

#### 10.336.3.2 const int VisualObjectMatch::getMatchCount () const

get count

Definition at line 158 of file VisualObject.C.

References [SortedLimitedList< KeypointMatch >::getSize\(\)](#).

Referenced by [InferoTemporal::input\(\)](#), [VisualObject::match\(\)](#), and [VisualObject-DB::matchObject\(\)](#).

**10.336.3.3 int VisualObjectMatch::getNumberKeyCompare () const**

Get the number of keypoint comparisons made.

Definition at line 226 of file VisualObject.C.

Referenced by InferoTemporal::input(), and VisualObjectDB::matchObject().

**10.336.3.4 const char \* VisualObjectMatch::getObjectName () const**

get object name

Definition at line 132 of file VisualObject.C.

Referenced by InferoTemporal::input().

**10.336.3.5 const Image< PixRGB< byte > > & VisualObjectMatch::getOutput () const**

get output

Definition at line 146 of file VisualObject.C.

**10.336.3.6 const float VisualObjectMatch::getScore () const**

get score

Definition at line 164 of file VisualObject.C.

Referenced by InferoTemporal::input().

**10.336.3.7 void VisualObjectMatch::incrementNumberKeyCompare (int *incr*)**

Increment the number of keypoint comparisons made.

Definition at line 220 of file VisualObject.C.

Referenced by VisualObject::match(), and VisualObjectDB::matchObject().

**10.336.3.8 void VisualObjectMatch::match (const Keypoint & *k1*, const Keypoint & *k2*)**

update output with a match

Definition at line 125 of file VisualObject.C.

References SortedLimitedList< KeypointMatch >::add().

Referenced by VisualObject::match().

#### **10.336.3.9 void VisualObjectMatch::setObjectName (char \* *name*)**

set object name

Definition at line 138 of file VisualObject.C.

Referenced by VisualObject::match().

#### **10.336.3.10 void VisualObjectMatch::setOutput (Image< PixRGB< byte > > & *img*)**

set output

Definition at line 152 of file VisualObject.C.

Referenced by VisualObject::match().

#### **10.336.3.11 void VisualObjectMatch::setOutputVerticalOffset (int *value*)**

Set the vertical offset of the output (height of the top image).

Definition at line 119 of file VisualObject.C.

Referenced by VisualObject::match().

#### **10.336.3.12 void VisualObjectMatch::writeOutput (char \* *outputFile*)**

write output

Definition at line 184 of file VisualObject.C.

References calculateScore(), Image< PixRGB< byte > >::drawLine(), KeypointMatch::getKeypoint1(), KeypointMatch::getKeypoint2(), SortedLimitedList< KeypointMatch >::getSize(), SortedLimitedList< KeypointMatch >::getValue(), and Raster::WriteRGB().

Referenced by InferoTemporal::input().

The documentation for this class was generated from the following files:

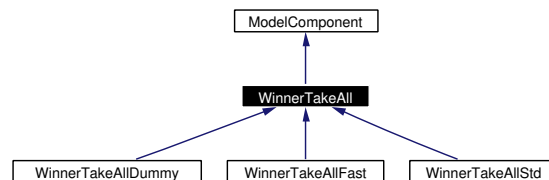
- [VisualObject.H](#)
- [VisualObject.C](#)

## 10.337 WinnerTakeAll Class Reference

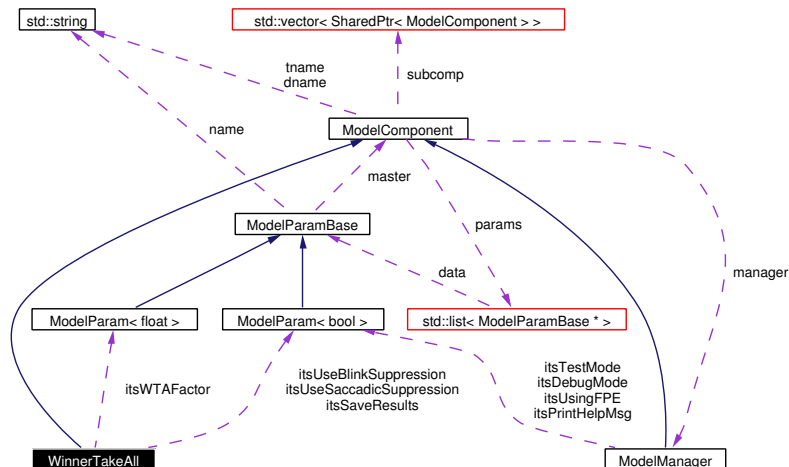
The winner-take-all base class.

```
#include <WinnerTakeAll.H>
```

Inheritance diagram for WinnerTakeAll:



Collaboration diagram for WinnerTakeAll:



## Public Member Functions

### Constructor, destructor, and reset

- `WinnerTakeAll` (`ModelManager` &mgr, const `std::string` &descr, Name="Winner-Take-All", const `std::string` &tagName="WinnerTakeAll")  
*Initialized constructor.*
- virtual void `exportOptions` (const int whatToExport, const bool recurse=true)

Export some of our [ModelParam](#) parameters as command-line-options.

- virtual [~WinnerTakeAll](#) ()  
*Destructor.*
- virtual void [reset](#) ()=0  
*Reset to initial state just after construction.*

### Input, evolve and output

- virtual void [input](#) (const [Image](#)< float > &in)=0  
*Set new input currents for all neurons.*
- virtual int [getWidth](#) () const=0  
*Get width of the network.*
- virtual int [getHeight](#) () const=0  
*Get height of the network.*
- virtual [Image](#)< float > [getV](#) (const bool normalize=false) const=0  
*Return all our membrane potential voltages as an [Image](#)<float>.*
- virtual float [getV](#) (const [Point2D](#) &p) const=0  
*Access function using [Point2D](#).*
- virtual void [evolve](#) (const double t, [Point2D](#) &winner)=0  
*Integrate inputs until time t (in s) and update membrane potentials.*

### Saccadic and blink suppression

- virtual void [saccadicSuppression](#) (const bool on)=0  
*Turn saccadic suppression on/off.*
- virtual void [blinkSuppression](#) (const bool on)=0  
*Turn blink suppression on/off.*

### Results

- virtual void [saveResults](#) (const [SharedPtr](#)< [OutputFrameSeries](#) > &ofs)  
*Save our various results.*



## Protected Attributes

- `ModelParam< bool > itsSaveResults`  
*Save our internals when `saveResults()` is called?*
- `ModelParam< float > itsWTAFactor`  
*Multiply WTA by a factor rather than normalize when doing `getV()`?*
- `ModelParam< bool > itsUseSaccadicSuppression`  
*use saccadic suppression?*
- `ModelParam< bool > itsUseBlinkSuppression`  
*use blink suppression?*

### 10.337.1 Detailed Description

The winner-take-all base class.

This is a 2D winner-take-all, a.k.a., spatial maximum detector. It is just a base class with virtual function definitions. Various winner-take-all variants are available below, which provide implementations for these virtual functions. The WTA receives input through the `input()` function, and its current state (membrane potential of neurons) can be retrieved using `getV()`. Various implementations will typically derive from class `Image` and inherit all the standard methods of class `Image`. The user has the responsibility of calling `evolve()` often to ensure that possible difference equations in the WTA's neurons run well.

Definition at line 59 of file `WinnerTakeAll.H`.

### 10.337.2 Constructor & Destructor Documentation

#### 10.337.2.1 `WinnerTakeAll::WinnerTakeAll (ModelManager & mgr, const std::string & descrName = "Winner-Take-All", const std::string & tagName = "WinnerTakeAll")`

Initialized constructor.

The SM will be resized and initialized the first time `input()` is called

Definition at line 50 of file `WinnerTakeAll.C`.

#### 10.337.2.2 `WinnerTakeAll::~WinnerTakeAll ()` [virtual]

Destructor.

Definition at line 60 of file WinnerTakeAll.C.

### 10.337.3 Member Function Documentation

#### 10.337.3.1 **virtual void WinnerTakeAll::blinkSuppression (const bool on)** [pure virtual]

Turn blink suppression on/off.

Implemented in [WinnerTakeAllStd](#), [WinnerTakeAllFast](#), and [WinnerTakeAllDummy](#).

#### 10.337.3.2 **virtual void WinnerTakeAll::evolve (const double t, [Point2D](#) & winner)** [pure virtual]

Integrate inputs until time t (in s) and update membrane potentials.

If new winner was selected, its coordinates are put in winner, otherwise, winner = (-1, -1).

Implemented in [WinnerTakeAllStd](#), [WinnerTakeAllFast](#), and [WinnerTakeAllDummy](#).

#### 10.337.3.3 **void WinnerTakeAll::exportOptions (const int *whatToExport*, const bool *recurse* = true)** [virtual]

Export some of our [ModelParam](#) parameters as command-line-options.

See the base function in [ModelComponent.H](#) for info. A base implementation is provided that will only export the base class options (see the [ModelParam](#) definitions in the protected section below).

Reimplemented from [ModelComponent](#).

Definition at line 64 of file WinnerTakeAll.C.

References [ModelComponent::exportOptions\(\)](#), [itsWTAFactor](#), [OPTEXP\\_CORE](#), [OPTEXP\\_SAVE](#), and [ModelManager::requestOption\(\)](#).

#### 10.337.3.4 **virtual int WinnerTakeAll::getHeight () const** [pure virtual]

Get height of the network.

Implemented in [WinnerTakeAllStd](#), [WinnerTakeAllFast](#), and [WinnerTakeAllDummy](#).

**10.337.3.5 virtual float WinnerTakeAll::getV (const [Point2D](#) & p) const**  
[pure virtual]

Access function using [Point2D](#).

Implemented in [WinnerTakeAllStd](#), [WinnerTakeAllFast](#), and [WinnerTakeAllDummy](#).

**10.337.3.6 virtual [Image](#)<float> WinnerTakeAll::getV (const bool *normalize* = false) const**  
[pure virtual]

Return all our membrane potential voltages as an [Image](#)<float>.

The current values are possibly normalized to 0..255. The normalization will use its-SalmapFactor if it is non-null

Implemented in [WinnerTakeAllStd](#), [WinnerTakeAllFast](#), and [WinnerTakeAllDummy](#).

Referenced by saveResults().

**10.337.3.7 virtual int WinnerTakeAll::getWidth () const** [pure virtual]

Get width of the network.

Implemented in [WinnerTakeAllStd](#), [WinnerTakeAllFast](#), and [WinnerTakeAllDummy](#).

**10.337.3.8 virtual void WinnerTakeAll::input (const [Image](#)< float > & in)**  
[pure virtual]

Set new input currents for all neurons.

This should initialize and resize the network if the network is currently uninitialized (e.g., just after construction or [reset\(\)](#)).

Implemented in [WinnerTakeAllStd](#), [WinnerTakeAllFast](#), and [WinnerTakeAllDummy](#).

**10.337.3.9 virtual void WinnerTakeAll::reset ()** [pure virtual]

Reset to initial state just after construction.

Implemented in [WinnerTakeAllStd](#), [WinnerTakeAllFast](#), and [WinnerTakeAllDummy](#).

**10.337.3.10 virtual void WinnerTakeAll::saccadicSuppression (const bool *on*)**  
[pure virtual]

Turn saccadic suppression on/off.

Implemented in [WinnerTakeAllStd](#), [WinnerTakeAllFast](#), and [WinnerTakeAllDummy](#).

### 10.337.3.11 `void WinnerTakeAll::saveResults (const SharedPtr< OutputFrameSeries > & ofs) [virtual]`

Save our various results.

A default implementation is provided, which just calls [getV\(\)](#) and saves that map with a file name prefix of "SM".

#### **Parameters:**

*ofs* will figure out file name/format and save our results.

Definition at line 81 of file WinnerTakeAll.C.

References [getV\(\)](#), and [ModelParam< bool >::getVal\(\)](#).

## 10.337.4 Member Data Documentation

### 10.337.4.1 `ModelParam<bool> WinnerTakeAll::itsSaveResults [protected]`

Save our internals when [saveResults\(\)](#) is called?

Definition at line 146 of file WinnerTakeAll.H.

### 10.337.4.2 `ModelParam<bool> WinnerTakeAll::itsUseBlinkSuppression [protected]`

use blink suppression?

Definition at line 155 of file WinnerTakeAll.H.

### 10.337.4.3 `ModelParam<bool> WinnerTakeAll::itsUseSaccadicSuppression [protected]`

use saccadic suppression?

Definition at line 152 of file WinnerTakeAll.H.

### 10.337.4.4 `ModelParam<float> WinnerTakeAll::itsWTAFactor [protected]`

Multiply WTA by a factor rather than normalize when doing [getV\(\)](#)?

Definition at line 149 of file WinnerTakeAll.H.

Referenced by [exportOptions\(\)](#).

The documentation for this class was generated from the following files:

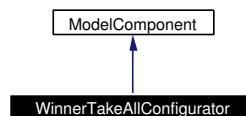
- [WinnerTakeAll.H](#)
- [WinnerTakeAll.C](#)

## 10.338 WinnerTakeAllConfigurator Class Reference

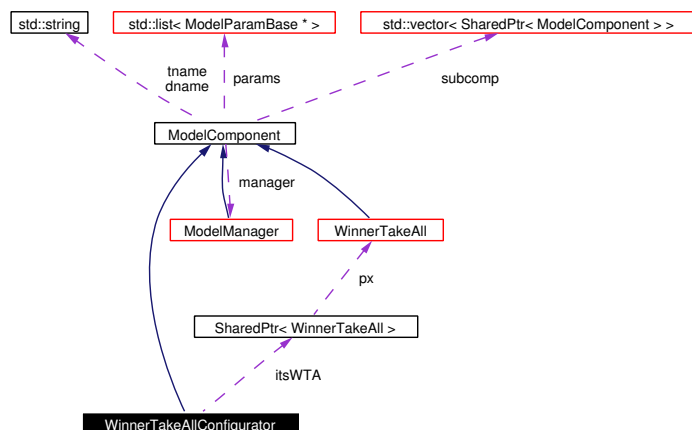
[WinnerTakeAll](#) configurator.

```
#include <WinnerTakeAll.H>
```

Inheritance diagram for WinnerTakeAllConfigurator:



Collaboration diagram for WinnerTakeAllConfigurator:



### Public Member Functions

- [WinnerTakeAllConfigurator](#) ([ModelManager](#) &mgr, const [std::string](#) &descr-Name="Winner-Take-All Configurator", const [std::string](#) &tagName="Winner-TakeAllConfigurator")

*Constructor.*

- virtual [~WinnerTakeAllConfigurator](#) ()

*destructor*

- virtual void [exportOptions](#) (const int whatToExport, const bool recurse=true)

*Export some of our [ModelParam](#) parameters as command-line-options.*

- [SharedPtr< WinnerTakeAll >](#) [getWTA \(\)](#) const

*Get the chosen WTA.*

## Protected Member Functions

- virtual bool [setModelParamString](#) (const [std::string](#) &name, const [std::string](#) &value, const bool recurse=false, const bool warn=true)

*Intercept people changing our [ModelParam](#).*

## Protected Attributes

- [ModelParam< std::string >](#) [itsWTAtype](#)

*type of winner-take-all*

### 10.338.1 Detailed Description

[WinnerTakeAll](#) configurator.

This will export the `-wta-type=XX` command-line option and will instantiate a WTA of the desired type as the option gets assigned a value. As this happens, new options may become available in the command-line. To see them, use `-help` AFTER you have chosen the type to use. The current WTA may be retrieved using [getWTA\(\)](#).

Definition at line 171 of file [WinnerTakeAll.H](#).

### 10.338.2 Constructor & Destructor Documentation

- 10.338.2.1 WinnerTakeAllConfigurator::WinnerTakeAllConfigurator**  
 ([ModelManager](#) & mgr, const [std::string](#) & descrName =  
 "Winner-Take-All Configurator", const [std::string](#) & tagName =  
 "WinnerTakeAllConfigurator")

Constructor.

Definition at line 94 of file [WinnerTakeAll.C](#).

References [OPTEXP\\_NONE](#).

### 10.338.2.2 **WinnerTakeAllConfigurator::~~WinnerTakeAllConfigurator ()** [virtual]

destructor

Definition at line 103 of file WinnerTakeAll.C.

## 10.338.3 Member Function Documentation

### 10.338.3.1 **void WinnerTakeAllConfigurator::exportOptions (const int *whatToExport*, const bool *recurse* = true) [virtual]**

Export some of our [ModelParam](#) parameters as command-line-options.

See the base function in [ModelComponent.H](#) for info. Actually, here we just keep a copy of *whatToExport* and we will call [exportOptions\(\)](#) on our WTA once we know its type and instantiate it.

Reimplemented from [ModelComponent](#).

Definition at line 107 of file WinnerTakeAll.C.

References [ModelComponent::exportOptions\(\)](#), [itsWTAType](#), [OPTEXP\\_CORE](#), and [ModelManager::requestOption\(\)](#).

### 10.338.3.2 **SharedPtr< WinnerTakeAll > WinnerTakeAllConfigurator::get- WTA () const**

Get the chosen WTA.

You should call this during [start\(\)](#) of the [ModelComponent](#) that needs the WTA. CAUTION: this may be a [SharedPtr\(NULL\)](#) if no WTA type has been chosen.

Definition at line 123 of file WinnerTakeAll.C.

### 10.338.3.3 **bool WinnerTakeAllConfigurator::setModelParamString (const [std::string](#) & *name*, const [std::string](#) & *value*, const bool *recurse* = false, const bool *warn* = true) [protected, virtual]**

Intercept people changing our [ModelParam](#).

See [ModelComponent.H](#); as parsing the command-line or reading a config file sets our name, we'll also here instantiate a controller of the proper type (and export its options)

Reimplemented from [ModelComponent](#).

Definition at line 127 of file WinnerTakeAll.C.

References [ModelComponent::addSubComponent\(\)](#), [SharedPtr< WinnerTakeAll](#)



>::get(), itsWTAtype, ModelComponent::removeSubComponent(), SharedPtr<WinnerTakeAll >::reset(), and ModelComponent::setModelParamString().

## 10.338.4 Member Data Documentation

### 10.338.4.1 [ModelParam<std::string>](#) [WinnerTakeAllConfigurator::its-WTAtype](#) [protected]

type of winner-take-all

Definition at line 199 of file WinnerTakeAll.H.

Referenced by exportOptions(), and setModelParamString().

The documentation for this class was generated from the following files:

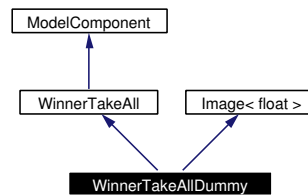
- [WinnerTakeAll.H](#)
- [WinnerTakeAll.C](#)

## 10.339 WinnerTakeAllDummy Class Reference

A dummy winner-take-all.

```
#include <WinnerTakeAll.H>
```

Inheritance diagram for WinnerTakeAllDummy:



Collaboration diagram for WinnerTakeAllDummy:

### Public Member Functions

- [WinnerTakeAllDummy](#) ([ModelManager](#) &mgr, const [std::string](#) &descr-Name="Winner-Take-All Dummy", const [std::string](#) &tagName="WinnerTake-AllDummy")  
*Initialized constructor.*
- virtual [~WinnerTakeAllDummy](#) ()  
*Destructor.*
- virtual void [reset](#) ()  
*Reset to initial state just after construction.*
- virtual void [input](#) (const [Image<float>](#) &in)  
*Set new input currents for all neurons.*
- virtual int [getWidth](#) () const  
*Get width of the network.*
- virtual int [getHeight](#) () const  
*Get height of the network.*
- virtual [Image<float>](#) [getV](#) (const bool normalize=false) const  
*Return all our membrane potential voltages as an [Image<float>](#).*

- virtual float [getV](#) (const [Point2D](#) &p) const

*Access function using [Point2D](#).*

- virtual void [evolve](#) (const double t, [Point2D](#) &winner)

*Integrate inputs until time t (in s) and update membrane potentials.*

- virtual void [saccadicSuppression](#) (const bool on)

*Turn saccadic suppression on/off.*

- virtual void [blinkSuppression](#) (const bool on)

*Turn blink suppression on/off.*

### 10.339.1 Detailed Description

A dummy winner-take-all.

This will never win and just ignores its inputs. Useful to avoid wasting time evolving a more complex WTA in cases where we don't care about the model attending to anywhere (e.g., when comparing saliency maps to human eye movements).

Definition at line 340 of file WinnerTakeAll.H.

### 10.339.2 Constructor & Destructor Documentation

#### 10.339.2.1 WinnerTakeAllDummy::WinnerTakeAllDummy ([ModelManager](#) &*mgr*, const [std::string](#) &*descrName* = "Winner-Take-All Dummy", const [std::string](#) &*tagName* = "WinnerTakeAllDummy")

Initialized constructor.

Definition at line 389 of file WinnerTakeAll.C.

#### 10.339.2.2 WinnerTakeAllDummy::~WinnerTakeAllDummy () [virtual]

Destructor.

Definition at line 396 of file WinnerTakeAll.C.

### 10.339.3 Member Function Documentation

#### 10.339.3.1 **void WinnerTakeAllDummy::blinkSuppression (const bool *on*)** [virtual]

Turn blink suppression on/off.

Implements [WinnerTakeAll](#).

Definition at line 444 of file WinnerTakeAll.C.

References `Image< float >::clear()`, and `ModelParam< bool >::getVal()`.

#### 10.339.3.2 **void WinnerTakeAllDummy::evolve (const double *t*, [Point2D](#) & *winner*)** [virtual]

Integrate inputs until time *t* (in s) and update membrane potentials.

Implements [WinnerTakeAll](#).

Definition at line 416 of file WinnerTakeAll.C.

References `Point2D::i`, and `Point2D::j`.

#### 10.339.3.3 **int WinnerTakeAllDummy::getHeight () const** [virtual]

Get height of the network.

Implements [WinnerTakeAll](#).

Definition at line 412 of file WinnerTakeAll.C.

References `Image< T >::getHeight()`.

#### 10.339.3.4 **float WinnerTakeAllDummy::getV (const [Point2D](#) & *p*) const** [virtual]

Access function using [Point2D](#).

Implements [WinnerTakeAll](#).

Definition at line 432 of file WinnerTakeAll.C.

References `Image< float >::getVal()`.

#### 10.339.3.5 **[Image](#)< float > WinnerTakeAllDummy::getV (const bool *normalize* = false) const** [virtual]

Return all our membrane potential voltages as an [Image](#)<float>.

Implements [WinnerTakeAll](#).

Definition at line 420 of file WinnerTakeAll.C.

References `ModelParam< float >::getVal()`, and `Image< T >::normalize()`.

#### 10.339.3.6 `int WinnerTakeAllDummy::getWidth () const [virtual]`

Get width of the network.

Implements [WinnerTakeAll](#).

Definition at line 408 of file WinnerTakeAll.C.

References `Image< T >::getWidth()`.

#### 10.339.3.7 `void WinnerTakeAllDummy::input (const Image< float > & in) [virtual]`

Set new input currents for all neurons.

This will initialize and resize the network if the network is currently uninitialized (e.g., just after construction or [reset\(\)](#)).

Implements [WinnerTakeAll](#).

Definition at line 404 of file WinnerTakeAll.C.

References `Image< float >::swap()`.

#### 10.339.3.8 `void WinnerTakeAllDummy::reset () [virtual]`

Reset to initial state just after construction.

Implements [WinnerTakeAll](#).

Definition at line 400 of file WinnerTakeAll.C.

References `Image< float >::freeMem()`.

#### 10.339.3.9 `void WinnerTakeAllDummy::saccadicSuppression (const bool on) [virtual]`

Turn saccadic suppression on/off.

Implements [WinnerTakeAll](#).

Definition at line 436 of file WinnerTakeAll.C.

References `Image< float >::clear()`, and `ModelParam< bool >::getVal()`.

The documentation for this class was generated from the following files:

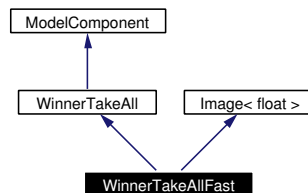
- [WinnerTakeAll.H](#)
- [WinnerTakeAll.C](#)

## 10.340 WinnerTakeAllFast Class Reference

A fast winner-take-all.

```
#include <WinnerTakeAll.H>
```

Inheritance diagram for WinnerTakeAllFast:



Collaboration diagram for WinnerTakeAllFast:

### Public Member Functions

- **WinnerTakeAllFast** (**ModelManager** &mgr, const **std::string** &descr-Name="Winner-Take-All Fast", const **std::string** &tagName="WinnerTake-AllFast")  
*Initialized constructor.*
- virtual **~WinnerTakeAllFast** ()  
*Destructor.*
- virtual void **reset** ()  
*Reset to initial state just after construction.*
- virtual void **input** (const **Image**< float > &in)  
*Set new input currents for all neurons.*
- virtual int **getWidth** () const  
*Get width of the network.*
- virtual int **getHeight** () const  
*Get height of the network.*
- virtual **Image**< float > **getV** (const bool normalize=false) const  
*Return all our membrane potential voltages as an **Image**<float>.*

- virtual float `getV` (const `Point2D` &p) const

*Access function using `Point2D`.*

- virtual void `evolve` (const double t, `Point2D` &winner)

*Integrate inputs until time t (in s) and update membrane potentials.*

- virtual void `saccadicSuppression` (const bool on)

*Turn saccadic suppression on/off.*

- virtual void `blinkSuppression` (const bool on)

*Turn blink suppression on/off.*

### 10.340.1 Detailed Description

A fast winner-take-all.

This just does an `Image<T>::findMax()` on the last input array received. It hence returns a winner each time `evolve()` is called.

Definition at line 290 of file `WinnerTakeAll.H`.

### 10.340.2 Constructor & Destructor Documentation

#### 10.340.2.1 `WinnerTakeAllFast::WinnerTakeAllFast (ModelManager & mgr, const std::string & descrName = "Winner-Take-All Fast", const std::string & tagName = "WinnerTakeAllFast")`

Initialized constructor.

The WTA will be resized and initialized the first time `input()` is called

Definition at line 317 of file `WinnerTakeAll.C`.

#### 10.340.2.2 `WinnerTakeAllFast::~WinnerTakeAllFast ()` [virtual]

Destructor.

Definition at line 324 of file `WinnerTakeAll.C`.



### 10.340.3 Member Function Documentation

#### 10.340.3.1 **void WinnerTakeAllFast::blinkSuppression (const bool on)** [virtual]

Turn blink suppression on/off.

Implements [WinnerTakeAll](#).

Definition at line 376 of file WinnerTakeAll.C.

References `Image< float >::clear()`, and `ModelParam< bool >::getVal()`.

#### 10.340.3.2 **void WinnerTakeAllFast::evolve (const double t, [Point2D](#) & winner)** [virtual]

Integrate inputs until time t (in s) and update membrane potentials.

Implements [WinnerTakeAll](#).

Definition at line 344 of file WinnerTakeAll.C.

References `Image< float >::findMax()`, `Point2D::i`, and `Point2D::j`.

#### 10.340.3.3 **int WinnerTakeAllFast::getHeight () const** [virtual]

Get height of the network.

Implements [WinnerTakeAll](#).

Definition at line 340 of file WinnerTakeAll.C.

References `Image< T >::getHeight()`.

#### 10.340.3.4 **float WinnerTakeAllFast::getV (const [Point2D](#) & p) const** [virtual]

Access function using [Point2D](#).

Implements [WinnerTakeAll](#).

Definition at line 364 of file WinnerTakeAll.C.

References `Image< float >::getVal()`.

#### 10.340.3.5 **[Image](#)< float > WinnerTakeAllFast::getV (const bool normalize = false) const** [virtual]

Return all our membrane potential voltages as an [Image<float>](#).

Implements [WinnerTakeAll](#).

Definition at line 352 of file WinnerTakeAll.C.

References `ModelParam< float >::getVal()`, and `Image< T >::normalize()`.

#### **10.340.3.6 int WinnerTakeAllFast::getWidth () const [virtual]**

Get width of the network.

Implements [WinnerTakeAll](#).

Definition at line 336 of file WinnerTakeAll.C.

References `Image< T >::getWidth()`.

#### **10.340.3.7 void WinnerTakeAllFast::input (const [Image](#)< float > & in) [virtual]**

Set new input currents for all neurons.

This will initialize and resize the network if the network is currently uninitialized (e.g., just after construction or [reset\(\)](#)).

Implements [WinnerTakeAll](#).

Definition at line 332 of file WinnerTakeAll.C.

References `Image< float >::swap()`.

#### **10.340.3.8 void WinnerTakeAllFast::reset () [virtual]**

Reset to initial state just after construction.

Implements [WinnerTakeAll](#).

Definition at line 328 of file WinnerTakeAll.C.

References `Image< float >::freeMem()`.

#### **10.340.3.9 void WinnerTakeAllFast::saccadicSuppression (const bool on) [virtual]**

Turn saccadic suppression on/off.

Implements [WinnerTakeAll](#).

Definition at line 368 of file WinnerTakeAll.C.

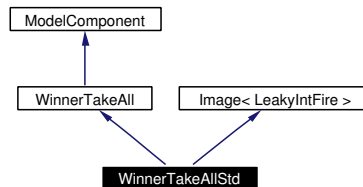
References `Image< float >::clear()`, and `ModelParam< bool >::getVal()`.

The documentation for this class was generated from the following files:

- [WinnerTakeAll.H](#)
- [WinnerTakeAll.C](#)

The standard winner-take-all.

Inheritance diagram for WinnerTakeAllStd:



- **WinnerTakeAllStd** (**ModelManager** &mgr, const **std::string** &descr-Name="Winner-Take-All Std", const **std::string** &>tagName="WinnerTakeAllStd")

- virtual  $\sim$ WinnerTakeAllStd ()

- virtual void **reset** ()

- virtual void `input` (const `Image`< float > &`in`)

Generated on Mon Nov 8 04:53:06 2004 for iLab Neuromorphic Vision Toolkit by Doxygen

- virtual int [getWidth](#) () const  
*Get width of the network.*
- virtual int [getHeight](#) () const  
*Get height of the network.*
- virtual [Image](#)< float > [getV](#) (const bool normalize=false) const  
*Return all our membrane potential voltages as an [Image](#)<float>.*
- virtual float [getV](#) (const [Point2D](#) &p) const  
*Access function using [Point2D](#).*
- virtual void [evolve](#) (const double t, [Point2D](#) &winner)  
*Integrate inputs until time t (in s) and update membrane potentials.*
- virtual void [saccadicSuppression](#) (const bool on)  
*Turn saccadic suppression on/off.*
- virtual void [blinkSuppression](#) (const bool on)  
*Turn blink suppression on/off.*

### 10.341.1 Detailed Description

The standard winner-take-all.

This is a trivial winner-take-all implementation, based on a 2D layer of [LeakyIntFire](#) neurons and a unique global inhibitory neuron. All neurons in the layer charge up in parallel; whenever one reaches threshold, it activates the inhibitory interneuron (which is connected to all neurons in the layer), which in turn resets all neurons in the layer. WinnerTakeAllStd is an [Image](#)<[LeakyIntFire](#)> and inherits all the standard methods of class [Image](#). To avoid confusion, we here add explicit [input\(\)](#) and [getV\(\)](#) methods (rather than providing conversion functions between [LeakyIntFire](#) and float, which could make the process more transparent but also possibly more confusing). The user has the responsibility of calling [evolve\(\)](#) often to ensure that the difference equations in the [LeakyIntegrator](#) neurons run well.

Definition at line 232 of file WinnerTakeAll.H.

## 10.341.2 Constructor & Destructor Documentation

### 10.341.2.1 WinnerTakeAllStd::WinnerTakeAllStd ([ModelManager](#) & mgr, const [std::string](#) & descrName = "Winner-Take-All Std", const [std::string](#) & tagName = "WinnerTakeAllStd")

Initialized constructor.

The WTA will be resized and initialized the first time [input\(\)](#) is called

Definition at line 176 of file WinnerTakeAll.C.

References [LeakyIntFire::setGleak\(\)](#).

### 10.341.2.2 WinnerTakeAllStd::~~WinnerTakeAllStd () [virtual]

Destructor.

Definition at line 187 of file WinnerTakeAll.C.

## 10.341.3 Member Function Documentation

### 10.341.3.1 void WinnerTakeAllStd::blinkSuppression (const bool on) [virtual]

Turn blink suppression on/off.

Implements [WinnerTakeAll](#).

Definition at line 304 of file WinnerTakeAll.C.

References [ModelParam< bool >::getVal\(\)](#).

### 10.341.3.2 void WinnerTakeAllStd::evolve (const double t, [Point2D](#) & winner) [virtual]

Integrate inputs until time t (in s) and update membrane potentials.

Implements [WinnerTakeAll](#).

Definition at line 226 of file WinnerTakeAll.C.

References [Image< float >::begin\(\)](#), [Image< LeakyIntFire >::beginw\(\)](#), [getHeight\(\)](#), [LeakyIntFire::getTimeStep\(\)](#), [getWidth\(\)](#), [Point2D::i](#), [LeakyIntFire::integrate\(\)](#), [Point2D::j](#), and [LeakyIntFire::setG\(\)](#).

**10.341.3.3 int WinnerTakeAllStd::getHeight () const [virtual]**

Get height of the network.

Implements [WinnerTakeAll](#).

Definition at line 222 of file WinnerTakeAll.C.

References [Image< T >::getHeight\(\)](#).

Referenced by [evolve\(\)](#).

**10.341.3.4 float WinnerTakeAllStd::getV (const [Point2D](#) & p) const [virtual]**

Access function using [Point2D](#).

Implements [WinnerTakeAll](#).

Definition at line 283 of file WinnerTakeAll.C.

References [LeakyIntFire::getV\(\)](#), and [Image< LeakyIntFire >::getVal\(\)](#).

**10.341.3.5 [Image](#)< float > WinnerTakeAllStd::getV (const bool *normalize* = false) const [virtual]**

Return all our membrane potential voltages as an [Image<float>](#).

The current values are possibly normalized to 0..255. The normalization will use its-SalmapFactor if it is non-null

Implements [WinnerTakeAll](#).

Definition at line 266 of file WinnerTakeAll.C.

References [Image< LeakyIntFire >::begin\(\)](#), [Image< T >::beginw\(\)](#), [Image< T >::endw\(\)](#), [Image< LeakyIntFire >::getDims\(\)](#), [ModelParam< float >::getVal\(\)](#), [NO\\_INIT](#), and [Image< T >::normalize\(\)](#).

**10.341.3.6 int WinnerTakeAllStd::getWidth () const [virtual]**

Get width of the network.

Implements [WinnerTakeAll](#).

Definition at line 218 of file WinnerTakeAll.C.

References [Image< T >::getWidth\(\)](#).

Referenced by [evolve\(\)](#).

**10.341.3.7 void WinnerTakeAllStd::input (const [Image](#)< float > & in)**  
[virtual]

Set new input currents for all neurons.

This will initialize and resize the network if the network is currently uninitialized (e.g., just after construction or [reset\(\)](#)).

Implements [WinnerTakeAll](#).

Definition at line 199 of file WinnerTakeAll.C.

References [Image](#)< [LeakyIntFire](#) >::beginw(), [Image](#)< [LeakyIntFire](#) >::endw(), [Image](#)< [LeakyIntFire](#) >::initialized(), [Image](#)< [LeakyIntFire](#) >::resize(), and [ZEROS](#).

**10.341.3.8 void WinnerTakeAllStd::reset ()** [virtual]

Reset to initial state just after construction.

Implements [WinnerTakeAll](#).

Definition at line 191 of file WinnerTakeAll.C.

References [Image](#)< float >::freeMem(), and [Image](#)< [LeakyIntFire](#) >::freeMem().

**10.341.3.9 void WinnerTakeAllStd::saccadicSuppression (const bool on)**  
[virtual]

Turn saccadic suppression on/off.

Implements [WinnerTakeAll](#).

Definition at line 296 of file WinnerTakeAll.C.

References [ModelParam](#)< bool >::getVal().

The documentation for this class was generated from the following files:

- [WinnerTakeAll.H](#)
- [WinnerTakeAll.C](#)

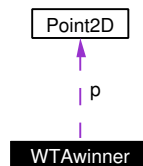


## 10.342 WTAwinner Class Reference

This is a an open class representing a WTA winner (covert attention shift).

```
#include <WTAwinner.H>
```

Collaboration diagram for WTAwinner:



### Public Member Functions

- [WTAwinner](#) ()  
*Default contructor, initializes to zeros.*
- [WTAwinner](#) (const [Point2D](#) &pp, const double tt, const double svv, const double avv, const bool bor)  
*Constructor given a [Point2D](#), time and other stuff.*
- [WTAwinner](#) (const int i, const int j, const double tt, const double svv, const double avv, const bool bor)  
*Constructor given (i, j), time and other stuff.*
- [Point2D](#) [getSMcoords](#) (const int smlevel) const  
*Return coordinates scaled down to scale of saliency map.*
- void [buildFromSMcoords](#) (const [Point2D](#) smcoords, const int smlevel, const bool useRandom, const double tt, const double svv, const double avv, const bool bor)  
*Build from coordinates given at the scale of the saliency map.*
- bool [isValid](#) () const  
*returns true if coordinates are not (-1, -1)*

### Public Attributes

- [Point2D](#) p

*The spatial coordinates.*

- double [t](#)

*The time, in seconds.*

- double [sv](#)

*The saliency map voltage at winning location, in Volts.*

- double [av](#)

*The Attention-Guidance-Map voltage at winning loc (Volts).*

- bool [boring](#)

*True is this shift was made out of boredom.*

### 10.342.1 Detailed Description

This is an open class representing a WTA winner (covert attention shift).

This class is an open (all members public) container for a [Point2D](#), time stamp, saliency voltage and possibly other things that characterize a WTA winner, or covert shift of attention. Like [Point2D](#), this class is fully inlined, so there is no WTAwinner.C file.

Definition at line 49 of file WTAwinner.H.

### 10.342.2 Constructor & Destructor Documentation

#### 10.342.2.1 WTAwinner::WTAwinner () [inline]

Default constructor, initializes to zeros.

Definition at line 87 of file WTAwinner.H.

#### 10.342.2.2 WTAwinner::WTAwinner (const [Point2D](#) & *pp*, const double *tt*, const double *svv*, const double *avv*, const bool *bor*) [inline]

Constructor given a [Point2D](#), time and other stuff.

Definition at line 92 of file WTAwinner.H.

#### 10.342.2.3 WTAwinner::WTAwinner (const int *i*, const int *j*, const double *tt*, const double *svv*, const double *avv*, const bool *bor*) [inline]

Constructor given (i, j), time and other stuff.

Definition at line 99 of file WTAwinner.H.

### 10.342.3 Member Function Documentation

**10.342.3.1** `void WTAwinner::buildFromSMcoords (const Point2D smcoords,  
const int smlevel, const bool useRandom, const double tt, const  
double svv, const double avv, const bool bor)` [inline]

Build from coordinates given at the scale of the saliency map.

Definition at line 115 of file WTAwinner.H.

References *av*, *boring*, [Point2D::i](#), [Point2D::j](#), and *sv*.

Referenced by `Brain::evolve()`.

**10.342.3.2** `Point2D WTAwinner::getSMcoords (const int smlevel) const`  
[inline]

Return coordinates scaled down to scale of saliency map.

Definition at line 106 of file WTAwinner.H.

References [Point2D::i](#), and [Point2D::j](#).

**10.342.3.3** `bool WTAwinner::isValid () const` [inline]

returns true if coordinates are not (-1, -1)

Definition at line 147 of file WTAwinner.H.

References [Point2D::i](#), and [Point2D::j](#).

Referenced by `ThresholdSaccadeController::checkPercepts()`, `SimulationViewerStd::drawFOA()`, and `SimulationViewerStd::linkFOAs()`.

### 10.342.4 Member Data Documentation

**10.342.4.1** `double WTAwinner::av`

The Attention-Guidance-Map voltage at winning loc (Volts).

Definition at line 80 of file WTAwinner.H.

Referenced by `buildFromSMcoords()`, and `SaccadeController::dumpQueues()`.

**10.342.4.2 bool WTAwinner::boring**

True is this shift was made out of boredom.

Definition at line 81 of file WTAwinner.H.

Referenced by SimulationViewerEyeMvt2::attentionShift(), buildFromSMcoords(), SimulationViewerStd::drawEye(), SimulationViewerStd::drawFOA(), SimulationViewerStd::drawHead(), Brain::evolve(), and VisualBuffer::input().

**10.342.4.3 Point2D WTAwinner::p**

The spatial coordinates.

Definition at line 77 of file WTAwinner.H.

Referenced by SimulationViewerEyeMvt2::attentionShift(), ThresholdSaccadeController::checkPercepts(), MonkeySaccadeController::computeWhenNewDecision(), TrivialSaccadeController::computeWhenNewDecision(), ModelWrapperSaccadeController::computeWhenNewPercept(), Monkey2SaccadeController::computeWhenNewPercept(), ThresholdFrictionSaccadeController::computeWhenNewPercept(), FrictionSaccadeController::computeWhenNewPercept(), ModelEyeHeadController::computeWhenResetPos(), MonkeySaccadeController::computeWhenResetPos(), MonkeyEyeHeadController::computeWhenResetPos(), ThresholdFrictionSaccadeController::computeWhenResetPos(), ThresholdSaccadeController::computeWhenResetPos(), FrictionSaccadeController::computeWhenResetPos(), SimulationViewerStd::drawFOA(), SaccadeController::dumpQueues(), FrictionSaccadeController::evolve(), Brain::evolve(), SimulationViewerStd::getTraj(), VisualBuffer::input(), SimulationViewerStd::linkFOAs(), SaccadeController::resetPos(), and FrictionSaccadeController::start1().

**10.342.4.4 double WTAwinner::sv**

The saliency map voltage at winning location, in Volts.

Definition at line 79 of file WTAwinner.H.

Referenced by buildFromSMcoords(), ThresholdSaccadeController::checkPercepts(), SaccadeController::dumpQueues(), and Brain::evolve().

**10.342.4.5 double WTAwinner::t**

The time, in seconds.

Definition at line 78 of file WTAwinner.H.

Referenced by MonkeySaccadeController::computeWhenNewDecision(), ModelWrapperSaccadeController::computeWhenNewPercept(), Monkey2Saccade-

Controller::computeWhenNewPercept(), FrictionSaccadeController::computeWhenNewPercept(), ModelEyeHeadController::computeWhenResetPos(), MonkeySaccadeController::computeWhenResetPos(), MonkeyEyeHeadController::computeWhenResetPos(), ThresholdFrictionSaccadeController::computeWhenResetPos(), FrictionSaccadeController::computeWhenResetPos(), SaccadeController::dumpQueues(), Brain::evolve(), and SaccadeController::resetPos().

The documentation for this class was generated from the following file:

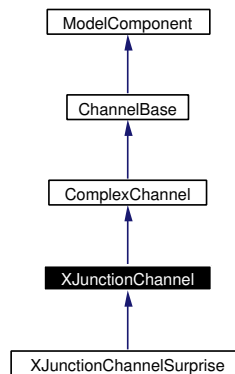
- [WTAwinner.H](#)

## 10.343 XJunctionChannel Class Reference

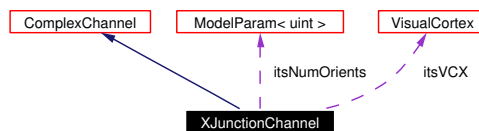
A X junction channel that detects crosses at different.

```
#include <Channels.H>
```

Inheritance diagram for XJunctionChannel:



Collaboration diagram for XJunctionChannel:



### Public Member Functions

- `XJunctionChannel` (`ModelManager` &mgr, `VisualCortex` \*vcx)  
*Construct with standard params.*
- virtual `~XJunctionChannel` ()  
*Destructor.*
- virtual void `exportOptions` (const int whatToExport, const bool recurse=true)  
*Export some of our `ModelParam` parameters as command-line-options.*
- virtual bool `setModelParamString` (const `std::string` &name, const `std::string` &value, const bool recurse=false, const bool warn=true)

*Overload so that we can reconfigure when our params get changed.*

## Protected Member Functions

- virtual void [buildSubChans](#) ()  
*(re-)build our subchannels*
- virtual void [doInput](#) (const [Image](#)< [PixRGB](#)< float > > \*, const [Image](#)< float > \*bwimg, const double t, const [Image](#)< byte > &clipMask)  
*Computes the cross junctions at different orientations.*

## Protected Attributes

- [ModelParam](#)< uint > [itsNumOrients](#)  
*number of junction subchannels*
- int [itsExportOpts](#)  
*keep track of what options to export in case we rebuild our subchans*
- [VisualCortex](#) \* [itsVCX](#)

### 10.343.1 Detailed Description

A X junction channel that detects crosses at different.

Definition at line 1364 of file Channels.H.

### 10.343.2 Constructor & Destructor Documentation

#### 10.343.2.1 XJunctionChannel::XJunctionChannel ([ModelManager](#) & mgr, [VisualCortex](#) \* vcx)

Construct with standard params.

Definition at line 2022 of file Channels.C.

References [buildSubChans\(\)](#), [OPTEXP\\_NONE](#), and [XJUNCTION](#).

**10.343.2.2 XJunctionChannel::~~XJunctionChannel () [virtual]**

Destructor.

Definition at line 2033 of file Channels.C.

**10.343.3 Member Function Documentation****10.343.3.1 void XJunctionChannel::buildSubChans () [protected, virtual]**

(re-)build our subchannels

Reimplemented in [XJunctionChannelSurprise](#).

Definition at line 2037 of file Channels.C.

References [ComplexChannel::addSubChan\(\)](#), [ModelParam< uint >::getVal\(\)](#), [makeShared\(\)](#), [ModelComponent::removeAllSubComponents\(\)](#), [ChannelBase::setHomogeneous\(\)](#), and [ChannelBase::visualFeature\(\)](#).

Referenced by [setModelParamString\(\)](#), and [XJunctionChannel\(\)](#).

**10.343.3.2 void XJunctionChannel::doInput (const [Image< PixRGB< float > > \\*](#), const [Image< float > \\*](#) *bwimg*, const double *t*, const [Image< byte > & clipMask](#)) [protected, virtual]**

Computes the cross junctions at different orientations.

Implements [ChannelBase](#).

Definition at line 2088 of file Channels.C.

References [ASSERT](#), [VisualCortex::hasChannel\(\)](#), [ChannelBase::input\(\)](#), [ComplexChannel::numChans\(\)](#), [ComplexChannel::subChan\(\)](#), and [uint](#).

**10.343.3.3 void XJunctionChannel::exportOptions (const int *whatToExport*, const bool *recurse* = true) [virtual]**

Export some of our [ModelParam](#) parameters as command-line-options.

See the base function in [ModelComponent.H](#) for info.

Reimplemented from [ComplexChannel](#).

Definition at line 2073 of file Channels.C.

References [ComplexChannel::exportOptions\(\)](#), [OPTEXP\\_CORE](#), and [ModelManager::requestOption\(\)](#).



**10.343.3.4** `bool XJunctionChannel::setModelParamString (const std::string & name, const std::string & value, const bool recurse = false, const bool warn = true) [virtual]`

Overload so that we can reconfigure when our params get changed.

Reimplemented from [ModelComponent](#).

Definition at line 2055 of file Channels.C.

References [buildSubChans\(\)](#), [ModelParamBase::getName\(\)](#), [ModelParam< uint >::getVal\(\)](#), [ComplexChannel::numChans\(\)](#), and [ModelComponent::setModelParamString\(\)](#).

## 10.343.4 Member Data Documentation

**10.343.4.1** `int XJunctionChannel::itsExportOpts [protected]`

keep track of what options to export in case we rebuild our subchans

Definition at line 1396 of file Channels.H.

**10.343.4.2** `ModelParam<uint> XJunctionChannel::itsNumOrients [protected]`

number of junction subchannels

Definition at line 1393 of file Channels.H.

The documentation for this class was generated from the following files:

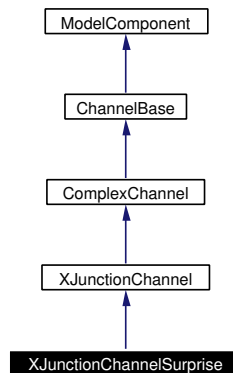
- [Channels.H](#)
- [Channels.C](#)

## 10.344 XJunctionChannelSurprise Class Reference

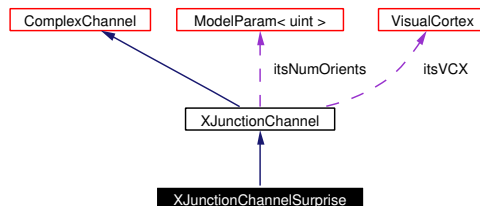
A X junction channel that detects crosses at different orientations.

```
#include <ChannelsSurprise.H>
```

Inheritance diagram for XJunctionChannelSurprise:



Collaboration diagram for XJunctionChannelSurprise:



### Public Member Functions

- `XJunctionChannelSurprise` (`ModelManager` &mgr, `VisualCortex` \*vcx)

*Construct with standard params.*

- `virtual ~XJunctionChannelSurprise ()`

*Destructor.*

## Protected Member Functions

- virtual void [buildSubChans](#) ()  
(re-)build our subchannels

### 10.344.1 Detailed Description

A X junction channel that detects crosses at different orientations.

Definition at line 596 of file ChannelsSurprise.H.

### 10.344.2 Constructor & Destructor Documentation

#### 10.344.2.1 XJunctionChannelSurprise::XJunctionChannelSurprise ([ModelManager](#) & *mgr*, [VisualCortex](#) \* *vcx*)

Construct with standard params.

Definition at line 841 of file ChannelsSurprise.C.

References [buildSubChans\(\)](#).

#### 10.344.2.2 XJunctionChannelSurprise::~~XJunctionChannelSurprise () [virtual]

Destructor.

Definition at line 849 of file ChannelsSurprise.C.

### 10.344.3 Member Function Documentation

#### 10.344.3.1 void XJunctionChannelSurprise::buildSubChans () [protected, virtual]

(re-)build our subchannels

Reimplemented from [XJunctionChannel](#).

Definition at line 853 of file ChannelsSurprise.C.

References [ComplexChannel::addSubChan\(\)](#), [ModelParam< uint >::getVal\(\)](#), [makeShared\(\)](#), [ModelComponent::removeAllSubComponents\(\)](#), [ChannelBase::setHomogeneous\(\)](#), and [ChannelBase::visualFeature\(\)](#).

Referenced by [XJunctionChannelSurprise\(\)](#).

The documentation for this class was generated from the following files:

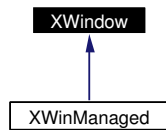
- [ChannelsSurprise.H](#)
- ChannelsSurprise.C

## 10.345 XWindow Class Reference

A simple X-window to display images.

```
#include <XWindow.H>
```

Inheritance diagram for XWindow:



### Public Member Functions

- [XWindow](#) (const int w=320, const int h=240, const int x=-1, const int y=-1, const char \*title="USC Neuromorphic Vision")  
*Constructor: create, display and map the window.*
- virtual [~XWindow](#) ()  
*Destructor.*
- void [setPosition](#) (const int x, const int y)  
*Set the position of an XWindow.*
- [Point2D](#) [getPosition](#) (void)  
*Get the position of an XWindow.*
- void [drawImage](#) (const [Image](#)< [PixRGB](#)< byte > > &img, const int x=0, const int y=0)  
*Draw an image into the XWindow.*
- void [drawImage](#) (const [Image](#)< byte > &img, const int x=0, const int y=0)  
*Draw an image into the XWindow.*
- virtual void [setTitle](#) (const char \*title)  
*set the title bar to a new string*
- [Dims](#) [getDims](#) () const  
*return the dimensions of this window*

## Protected Member Functions

- void **createImage** (const int w, const int h)
- void **destroyImage** ()

## Protected Attributes

- int **width**
- int **height**
- Display \* **display**
- int **screen**
- int **depth**
- Window **window**
- XImage \* **image**
- Visual \* **visual**
- GC **gc**
- XShmSegmentInfo **shminfo**
- char \* **buffer**
- bool **shm\_buffer**
- int **iw**
- int **ih**
- int **ix**
- int **iy**
- int **byt\_per\_pix**

### 10.345.1 Detailed Description

A simple X-window to display images.

Use this class to create an window on your screen and display an image.

It will use shared memory between us and the X server if supported by the X server and the display is local. Resizing is not allowed; so any image passed to [drawImage\(\)](#) must have dimensions that match those specified at onstruction of the XWindow.

Definition at line 57 of file XWindow.H.

### 10.345.2 Constructor & Destructor Documentation

#### 10.345.2.1 XWindow::XWindow (const int w = 320, const int h = 240, const int x = -1, const int y = -1, const char \* title = "USC Neuromorphic Vision")

Constructor: create, display and map the window.

**Parameters:**

- w** Width of usable image area.
- h** height of usable image area.
- x** Horizontal position hint for window placement; use -1 for automatic placement.
- y** Vertical position hint for window placement; use -1 for automatic placement.
- title** String that will appear in the window title.

Definition at line 57 of file XWindow.C.

References `setPosition()`, and `setTitle()`.

**10.345.2.2 XWindow::~XWindow () [virtual]**

Destructor.

Definition at line 274 of file XWindow.C.

**10.345.3 Member Function Documentation****10.345.3.1 void XWindow::drawImage (const Image< byte > &img, const int x = 0, const int y = 0)**

Draw an image into the XWindow.

**Parameters:**

- img** The image to draw. Image size must match the usable image area specified at construction of the XWindow.
- img** is the image to display
- x** if the x coordinate where the image should start in the window.
- y** if the y coordinate where the image should start in the window.

Reimplemented in [XWinManaged](#).

Definition at line 231 of file XWindow.C.

References `ASSERT`, `Image< T >::getArrayPtr()`, `Image< T >::getHeight()`, and `Image< T >::getWidth()`.

**10.345.3.2 void XWindow::drawImage (const Image< PixRGB< byte > > &img, const int x = 0, const int y = 0)**

Draw an image into the XWindow.

**Parameters:**

**img** The image to draw. [Image](#) size must match the usable image area specified at construction of the XWindow.

**img** is the image to display

**x** if the x coordinate where the image should start in the window.

**y** if the y coordinate where the image should start in the window.

Reimplemented in [XWinManaged](#).

Definition at line 179 of file XWindow.C.

References ASSERT.

Referenced by XWinManaged::drawImage(), main(), and XWinManaged::XWinManaged().

**10.345.3.3 [Dims](#) XWindow::getDims () const**

return the dimensions of this window

Definition at line 337 of file XWindow.C.

Referenced by MbariResultViewer::displayImage().

**10.345.3.4 [Point2D](#) XWindow::getPosition (void)**

Get the position of an XWindow.

Definition at line 171 of file XWindow.C.

**10.345.3.5 void XWindow::setPosition (const int x, const int y)**

Set the position of an XWindow.

**Parameters:**

**x** Horizontal position hint for window placement

**y** Vertical position hint for window placement

Definition at line 165 of file XWindow.C.

Referenced by XWindow().

**10.345.3.6 void XWindow::setTitle (const char \* title) [virtual]**

set the title bar to a new string



Definition at line 154 of file XWindow.C.

Referenced by MbariResultViewer::displayImage(), main(), and XWindow().

The documentation for this class was generated from the following files:

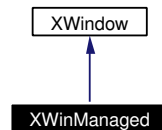
- [XWindow.H](#)
- [XWindow.C](#)

## 10.346 XWinManaged Class Reference

A window class with active window management.

```
#include <XWinManaged.H>
```

Inheritance diagram for XWinManaged:



Collaboration diagram for XWinManaged:

### Public Member Functions

- [XWinManaged](#) (const int w=320, const int h=240, const int x=-1, const int y=-1, const char \*title="USC Neuromorphic Vision")  
*Constructor: create, display and map the window, start the event handling.*
- [XWinManaged](#) (const [Image](#)< [PixRGB](#)< [byte](#) > > &img, const char \*title="USC Neuromorphic Vision")  
*Constructor: create a window as large as img and display img.*
- [XWinManaged](#) (const [Image](#)< [byte](#) > &img, const char \*title="USC Neuromorphic Vision")  
*Constructor: create a window as large as img and display img.*
- [XWinManaged](#) (const [Image](#)< [float](#) > &img, const char \*title="USC Neuromorphic Vision", bool normalize=true)  
*Constructor: create a window as large as img and display img.*
- virtual [~XWinManaged](#) ()  
*Destructor: destroy the window and stop the event handling.*
- void [drawImage](#) (const [Image](#)< [float](#) > &img, const int x=0, const int y=0, bool normalize=true)  
*Draw an image into the [XWindow](#).*
- void [drawImage](#) (const [Image](#)< [PixRGB](#)< [byte](#) > > &img, const int x=0, const int y=0)

*Draw an image into the [XWindow](#).*

- void [drawImage](#) (const [Image](#)< [byte](#) > &img, const int x=0, const int y=0)  
*Draw an image into the [XWindow](#).*
- bool [pressedCloseButton](#) ()  
*returns whether the user has clicked the close button.*
- [Point2D](#) [getLastMouseClicked](#) ()
- int [getLastKeyPress](#) ()

## Protected Member Functions

- void [initHandling](#) (void)  
*initialize the event handling*
- void [event\\_loop](#) (void)  
*loop that periodically checks for events*
- virtual void [event\\_handler](#) (XEvent \*event)  
*deal with the events coming in - feel free to override*
- virtual void [redraw](#) ()  
*redraw the image - may be overridden*

## Static Protected Member Functions

- void [event\\_loop\\_](#) ([XWinManaged](#) \*thisobj)

## Protected Attributes

- Atom [deleteAtom](#)
- pthread\_t [eventLoopThread](#)
- bool [windowPresent](#)
- bool [toBeClosed](#)
- [std::deque](#)< int > [lastKey](#)
- [std::deque](#)< [Point2D](#) > [lastPoint](#)

## Static Protected Attributes

- const unsigned int [queueLen](#) = 5

### 10.346.1 Detailed Description

A window class with active window management.

This class is an extension of the [XWindow](#) class to allow event handling. In the current implementation, *exposure* and *resize* events are being handled. Instead of destroying the window, clicking the *close* button will set a flag that can be accessed by calling [pressedCloseButton\(\)](#).

Definition at line 61 of file XWinManaged.H.

### 10.346.2 Constructor & Destructor Documentation

#### 10.346.2.1 XWinManaged::XWinManaged (const int *w* = 320, const int *h* = 240, const int *x* = -1, const int *y* = -1, const char \* *title* = "USC Neuromorphic Vision")

Constructor: create, display and map the window, start the event handling.

The call of to the constructor is identical to the parent class [XWindow](#). In addition to creating the window, the event handling is being started.

##### Parameters:

- w*** Width of usable image area.
- h*** height of usable image area.
- x*** Horizontal position hint for window placement; use -1 for automatic placement.
- y*** Vertical position hint for window placement; use -1 for automatic placement.
- title*** String that will appear in the window title.

Definition at line 70 of file XWinManaged.C.

References [initHandling\(\)](#).

#### 10.346.2.2 XWinManaged::XWinManaged (const [Image](#)< [PixRGB](#)< [byte](#) > > & *img*, const char \* *title* = "USC Neuromorphic Vision")

Constructor: create a window as large as *img* and display *img*.

Definition at line 80 of file XWinManaged.C.

References [XWindow::drawImage\(\)](#), and [initHandling\(\)](#).

#### 10.346.2.3 XWinManaged::XWinManaged (const [Image](#)< [byte](#) > & *img*, const char \* *title* = "USC Neuromorphic Vision")

Constructor: create a window as large as *img* and display *img*.

Definition at line 90 of file XWinManaged.C.

References XWindow::drawImage(), and initHandling().

#### 10.346.2.4 XWinManaged::XWinManaged (const Image< float > & img, const char \* title = "USC Neuromorphic Vision", bool normalize = true)

Constructor: create a window as large as img and display img.

Definition at line 99 of file XWinManaged.C.

References drawImage(), and initHandling().

#### 10.346.2.5 XWinManaged::~XWinManaged () [virtual]

Destructor: destroy the window and stop the event handling.

Definition at line 110 of file XWinManaged.C.

### 10.346.3 Member Function Documentation

#### 10.346.3.1 void XWinManaged::drawImage (const Image< byte > & img, const int x = 0, const int y = 0)

Draw an image into the XWindow.

##### Parameters:

*img* The image to draw. Image size must match the usable image area specified at construction of the XWindow.

*img* is the image to display

*x* if the x coordinate where the image should start in the window.

*y* if the y coordinate where the image should start in the window.

Reimplemented from XWindow.

Definition at line 141 of file XWinManaged.C.

References XWindow::drawImage().

#### 10.346.3.2 void XWinManaged::drawImage (const Image< PixRGB< byte > > & img, const int x = 0, const int y = 0)

Draw an image into the XWindow.

**Parameters:**

- img* The image to draw. [Image](#) size must match the usable image area specified at construction of the [XWindow](#).
- img* is the image to display
- x* if the x coordinate where the image should start in the window.
- y* if the y coordinate where the image should start in the window.

Reimplemented from [XWindow](#).

Definition at line 134 of file XWinManaged.C.

References [XWindow::drawImage\(\)](#).

### 10.346.3.3 void XWinManaged::drawImage (const [Image](#)< float > & *img*, const int *x* = 0, const int *y* = 0, bool *normalize* = true)

Draw an image into the [XWindow](#).

**Parameters:**

- img* The image to draw. [Image](#) size must match the usable image area specified at construction of the [XWindow](#).
- img* is the image to display
- x* if the x coordinate where the image should start in the window.
- y* if the y coordinate where the image should start in the window.

Definition at line 119 of file XWinManaged.C.

References [Image< PixRGB< float > >::clamp\(\)](#), [XWindow::drawImage\(\)](#), [image](#), and [Image< PixRGB< float > >::normalize\(\)](#).

Referenced by [MbariResultViewer::displayImage\(\)](#), [main\(\)](#), and [XWinManaged\(\)](#).

### 10.346.3.4 void XWinManaged::event\_handler (XEvent \* *event*) [protected, virtual]

deal with the events coming in - feel free to override

Definition at line 171 of file XWinManaged.C.

References [redraw\(\)](#).

Referenced by [event\\_loop\(\)](#).

**10.346.3.5 void XWinManaged::event\_loop (void) [protected]**

loop that periodically checks for events

Definition at line 148 of file XWinManaged.C.

References event\_handler().

**10.346.3.6 int XWinManaged::getLastKeyPress ()**

returns the key press at the front of the keyboard buffer if no key press has occurred since the last call returns -1

Definition at line 247 of file XWinManaged.C.

References redraw().

**10.346.3.7 [Point2D](#) XWinManaged::getLastMouseClicked ()**

returns the last mouse click at the front of the click buffer does not distinguish between right and left buttons if no mouse clicks have occurred since the last call returns (-1,-1)

Definition at line 260 of file XWinManaged.C.

References redraw().

**10.346.3.8 void XWinManaged::initHandling (void) [protected]**

initialize the event handling

Definition at line 53 of file XWinManaged.C.

Referenced by XWinManaged().

**10.346.3.9 bool XWinManaged::pressedCloseButton () [inline]**

returns whether the user has clicked the *close* button.

Definition at line 172 of file XWinManaged.H.

Referenced by main().

**10.346.3.10 void XWinManaged::redraw () [protected, virtual]**

redraw the image - may be overridden

Definition at line 230 of file XWinManaged.C.

Referenced by `event_handler()`, `getLastKeyPress()`, and `getLastMouseClicked()`.

The documentation for this class was generated from the following files:

- [XWinManaged.H](#)
- [XWinManaged.C](#)

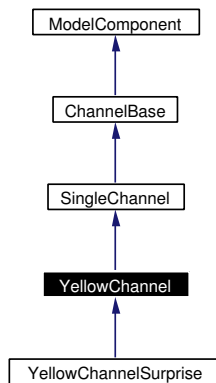


## 10.347 YellowChannel Class Reference

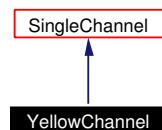
A yellow double-opponent color channel.

```
#include <Channels.H>
```

Inheritance diagram for YellowChannel:



Collaboration diagram for YellowChannel:



### Public Member Functions

- [YellowChannel](#) ([ModelManager](#) &mgr)  
*Construct with standard params.*
- virtual [~YellowChannel](#) ()  
*Destructor.*

### 10.347.1 Detailed Description

A yellow double-opponent color channel.

Definition at line 219 of file Channels.H.

## 10.347.2 Constructor & Destructor Documentation

### 10.347.2.1 YellowChannel::YellowChannel ([ModelManager](#) & *mgr*)

Construct with standard params.

Definition at line 265 of file Channels.C.

References [makeShared\(\)](#), [ModelParam< bool >::setVal\(\)](#), and [Y](#).

### 10.347.2.2 YellowChannel::~[~YellowChannel](#) () [virtual]

Destructor.

Definition at line 272 of file Channels.C.

The documentation for this class was generated from the following files:

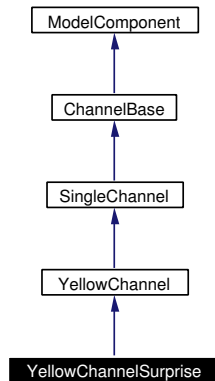
- [Channels.H](#)
- [Channels.C](#)

## 10.348 YellowChannelSurprise Class Reference

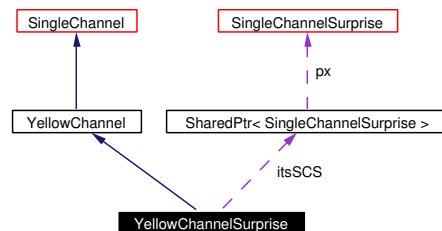
A yellow double-opponent color channel.

```
#include <ChannelsSurprise.H>
```

Inheritance diagram for YellowChannelSurprise:



Collaboration diagram for YellowChannelSurprise:



### Public Member Functions

- `YellowChannelSurprise (ModelManager &mgr)`  
*Construct with standard params.*
- `virtual ~YellowChannelSurprise ()`  
*Destructor.*
- `virtual Image< float > computeSubmap (const uint index) const`  
*computeSubmap() is special and will return surprise*

## Protected Attributes

- [SharedPtr](#)< [SingleChannelSurprise](#) > [itsSCS](#)  
*our surprise computer*

### 10.348.1 Detailed Description

A yellow double-opponent color channel.

Definition at line 249 of file ChannelsSurprise.H.

### 10.348.2 Constructor & Destructor Documentation

#### 10.348.2.1 [YellowChannelSurprise::YellowChannelSurprise](#) ([ModelManager](#) & *mgr*)

Construct with standard params.

Definition at line 229 of file ChannelsSurprise.C.

References [ModelComponent::addSubComponent\(\)](#).

#### 10.348.2.2 [YellowChannelSurprise::~~YellowChannelSurprise](#) () [virtual]

Destructor.

Definition at line 235 of file ChannelsSurprise.C.

### 10.348.3 Member Function Documentation

#### 10.348.3.1 [Image](#)< float > [YellowChannelSurprise::computeSubmap](#) (const [uint index](#)) const [virtual]

[computeSubmap\(\)](#) is special and will return surprise

Reimplemented from [SingleChannel](#).

Definition at line 238 of file ChannelsSurprise.C.

References [uint](#).

## 10.348.4 Member Data Documentation

### 10.348.4.1 [SharedPtr<SingleChannelSurprise> YellowChannelSurprise::its-SCS](#) [protected]

our surprise computer

Definition at line 262 of file ChannelsSurprise.H.

The documentation for this class was generated from the following files:

- [ChannelsSurprise.H](#)
- ChannelsSurprise.C



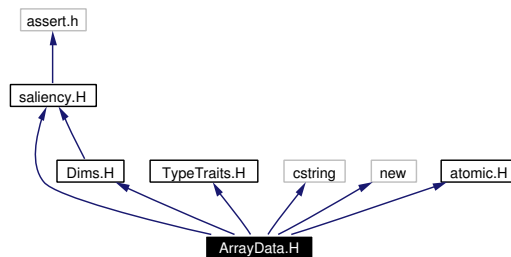
## Chapter 11

# iLab Neuromorphic Vision Toolkit File Documentation

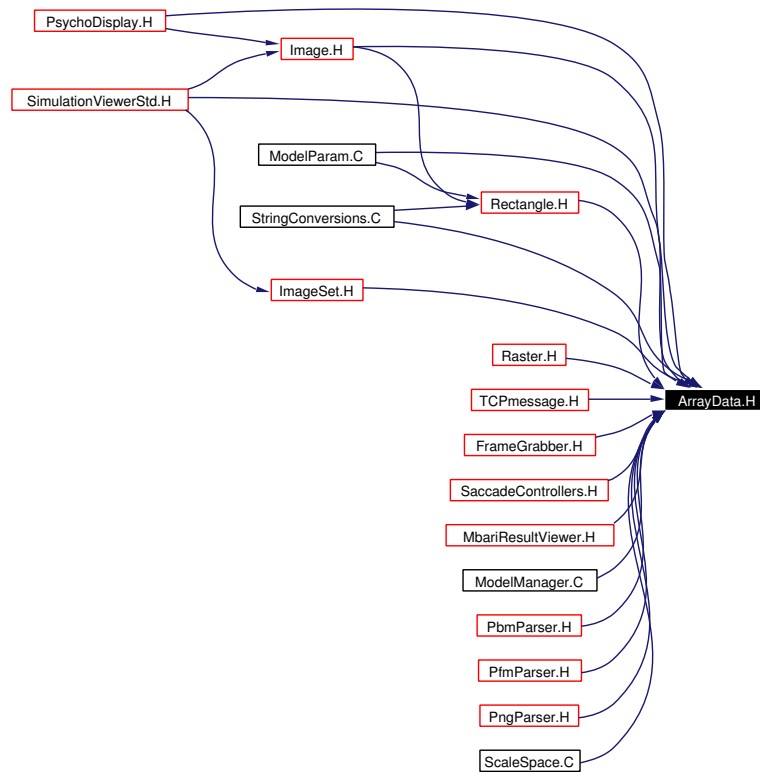
### 11.1 ArrayData.H File Reference

```
#include "saliency.H"  
#include "TypeTraits.H"  
#include "Dims.H"  
#include <cstring>  
#include <new>  
#include "atomic.H"
```

Include dependency graph for ArrayData.H:



This graph shows which files directly or indirectly include this file:



## Classes

- class [ArrayData](#)

*Implements a ref-counted 2-D array, should only be used via [ArrayHandle](#).*

- class [ArrayHandle](#)

*This class provides ref-counting and copy-on-write for [ArrayData](#)'s.*

- struct [ArrayHelper](#)

*This is a helper template struct for [ArrayData](#)'s constructors.*

- struct [ArrayHelper< T, true >](#)

*Specialization of [ArrayHelper](#) for builtin/trivial types.*



## Defines

- `#define REFCOUNT_THREADSAFE`
- `#define RC_SET(x, v) atomic_set(&x, v)`  
*Set ref-count x to the given value v.*
- `#define RC_INCR(x) atomic_inc(&x)`  
*Increment the ref-count x by one.*
- `#define RC_DECR_TEST_ZERO(x) bool(atomic_dec_and_test(&x))`  
*Decrement the ref-count x by one; return true if the new value is zero.*
- `#define RC_GET(x) atomic_read(&x)`  
*Get the integral value of the ref-count x.*

## Typedefs

- `typedef atomic_int_t RefCountType`  
*In threadsafe mode, the RefCountType is an integral type with atomic operations.*

## Enumerations

- `enum StoragePolicy { MAKE_OWN_COPY, WRITE_THRU }`  
*Specifies what to do with the T\* used to init an [ArrayData](#).*
- `enum InitPolicy { ZEROS, NO_INIT }`  
*Specifies how to initialize the pixels of a new image.*

### 11.1.1 Detailed Description

The guts of the 2D ref-counted copy-on-write array implementation  
Definition in file [ArrayData.H](#).

### 11.1.2 Define Documentation

#### 11.1.2.1 `#define RC_DECR_TEST_ZERO(x) bool(atomic_dec_and_test(&x))`

Decrement the ref-count x by one; return true if the new value is zero.

Definition at line 69 of file ArrayData.H.

Referenced by ArrayData< T >::release().

#### 11.1.2.2 `#define RC_GET(x) atomic_read(&x)`

Get the integral value of the ref-count *x*.

Definition at line 73 of file ArrayData.H.

Referenced by ArrayData< T >::isShared(), and ArrayData< T >::refCount().

#### 11.1.2.3 `#define RC_INCR(x) atomic_inc(&x)`

Increment the ref-count *x* by one.

Definition at line 67 of file ArrayData.H.

Referenced by ArrayData< T >::acquire().

#### 11.1.2.4 `#define RC_SET(x, v) atomic_set(&x, v)`

Set ref-count *x* to the given value *v*.

Definition at line 65 of file ArrayData.H.

Referenced by ArrayData< T >::ArrayData().

### 11.1.3 Typedef Documentation

#### 11.1.3.1 `typedef atomic_int_t RefCountType`

In threadsafe mode, the RefCountType is an integral type with atomic operations.

Definition at line 63 of file ArrayData.H.

### 11.1.4 Enumeration Type Documentation

#### 11.1.4.1 `enum InitPolicy`

Specifies how to initialize the pixels of a new image.

##### Enumeration values:

**ZEROS** All elements are initialized to the pixel type's zero or default value.

***NO\_INIT*** Pixels are only minimally initialized.

For trivial types, this means the pixel values are not initialized at all; for non-trivial user-defined types, this means each pixel is copy-constructed from the type's default value.

Definition at line 101 of file ArrayData.H.

#### 11.1.4.2 enum [StoragePolicy](#)

Specifies what to do with the T\* used to init an [ArrayData](#).

**Enumeration values:**

***MAKE\_OWN\_COPY*** Make an initial copy of the data, then copy-on-write as usual.

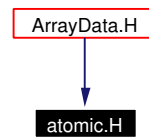
***WRITE\_THRU*** Borrow the data, with write-through.

Writing to the Array will modify the T[] that initialized it.

Definition at line 90 of file ArrayData.H.

## 11.2 atomic.H File Reference

This graph shows which files directly or indirectly include this file:



### Classes

- struct **atomic\_int\_t**

### Defines

- #define **LOCK** "lock ; "
- #define **atomic\_read**(v) ((v) → counter)
- #define **atomic\_set**(v, i) (((v) → counter) = (i))

### Functions

- \_\_inline\_\_ void **atomic\_inc** (atomic\_int\_t \*v)
- \_\_inline\_\_ int **atomic\_dec\_and\_test** (atomic\_int\_t \*v)

#### 11.2.1 Detailed Description

Inline x86 assembly functions imported from Linux kernel headers circa version 2.4.18-41mdk.

Definition in file [atomic.H](#).

#### 11.2.2 Define Documentation

##### 11.2.2.1 #define atomic\_read(v) ((v) → counter)

**atomic\_read** - read atomic variable : pointer of type `atomic_int_t`

Atomically reads the value of . Note that the guaranteed useful range of an `atomic_int_t` is only 24 bits.

Definition at line 31 of file `atomic.H`.

### 11.2.2.2 **#define atomic\_set(v, i) (((v) → counter) = (i))**

atomic\_set - set atomic variable : pointer of type atomic\_int\_t : required value

Atomically sets the value of to . Note that the guaranteed useful range of an atomic\_int\_t is only 24 bits.

Definition at line 41 of file atomic.H.

## 11.2.3 Function Documentation

### 11.2.3.1 **\_\_inline\_\_ int atomic\_dec\_and\_test (atomic\_int\_t \* v) [static]**

atomic\_dec\_and\_test - decrement and test : pointer of type atomic\_int\_t

Atomically decrements by 1 and returns true if the result is 0, or false for all other cases. Note that the guaranteed useful range of an atomic\_int\_t is only 24 bits.

Definition at line 67 of file atomic.H.

### 11.2.3.2 **\_\_inline\_\_ void atomic\_inc (atomic\_int\_t \* v) [static]**

atomic\_inc - increment atomic variable : pointer of type atomic\_int\_t

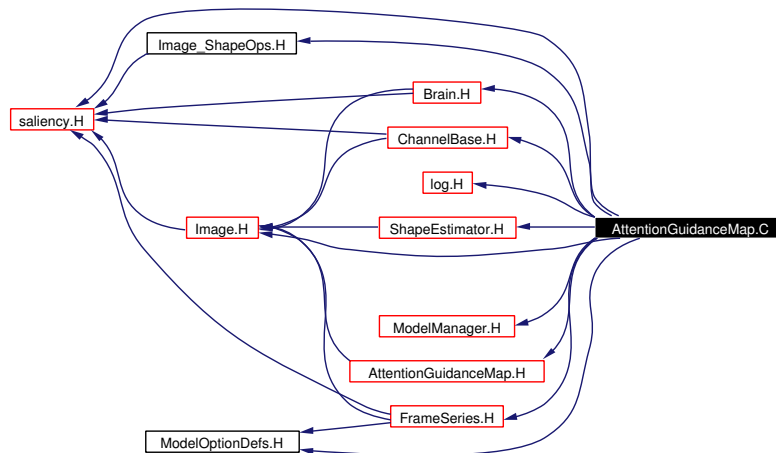
Atomically increments by 1. Note that the guaranteed useful range of an atomic\_int\_t is only 24 bits.

Definition at line 50 of file atomic.H.

## 11.3 AttentionGuidanceMap.C File Reference

```
#include "saliency.H"
#include "Image.H"
#include "Image_ShapeOps.H"
#include "AttentionGuidanceMap.H"
#include "log.H"
#include "ChannelBase.H"
#include "ModelOptionDefs.H"
#include "ModelManager.H"
#include "FrameSeries.H"
#include "ShapeEstimator.H"
#include "Brain.H"
```

Include dependency graph for AttentionGuidanceMap.C:



### 11.3.1 Detailed Description

Implementation for task-relevance map class

Definition in file [AttentionGuidanceMap.C](#).

## 11.4 AttentionGuidanceMap.H File Reference

```
#include "Image.H"
```

```
#include "ModelComponent.H"
```

Include dependency graph for AttentionGuidanceMap.H:

This graph shows which files directly or indirectly include this file:

### Classes

- class [AttentionGuidanceMap](#)  
*The task-relevance map base class.*
- class [AttentionGuidanceMapConfigurator](#)  
*AttentionGuidanceMap configurator.*
- class [AttentionGuidanceMapStd](#)  
*The standard task-relevance map.*

### 11.4.1 Detailed Description

Class declarations for task-relevance map class

Definition in file [AttentionGuidanceMap.H](#).

## 11.5 AudioGrabber.C File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#include <sys/soundcard.h>
#include <sys/ioctl.h>
#include "saliency.H"
#include "log.H"
#include "AudioGrabber.H"
#include "ModelManager.H"
#include "ModelOptionDefs.H"
Include dependency graph for AudioGrabber.C:
```

### 11.5.1 Detailed Description

Grab audio samples from /dev/dsp

Definition in file [AudioGrabber.C](#).

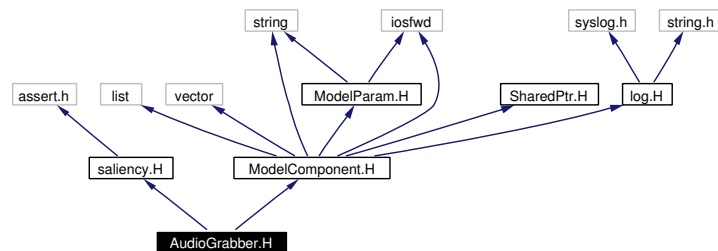


## 11.6 AudioGrabber.H File Reference

```
#include "saliency.H"
```

```
#include "ModelComponent.H"
```

Include dependency graph for AudioGrabber.H:



This graph shows which files directly or indirectly include this file:



### Classes

- class [AudioGrabber](#)  
*Class to grab audio samples from an audio device.*

#### 11.6.1 Detailed Description

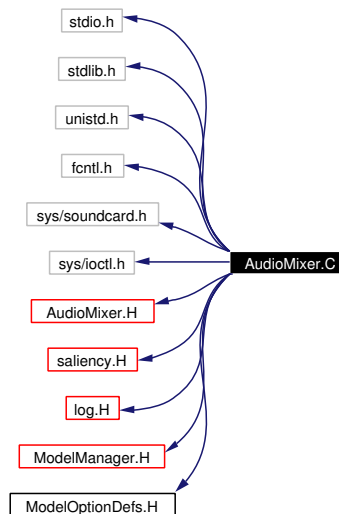
Grab audio samples from /dev/dsp

Definition in file [AudioGrabber.H](#).

## 11.7 AudioMixer.C File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#include <sys/soundcard.h>
#include <sys/ioctl.h>
#include "AudioMixer.H"
#include "saliency.H"
#include "log.H"
#include "ModelManager.H"
#include "ModelOptionDefs.H"
```

Include dependency graph for AudioMixer.C:



### 11.7.1 Detailed Description

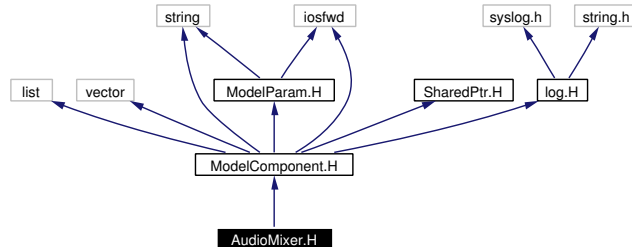
A simple interface to select audio recording source

Definition in file [AudioMixer.C](#).

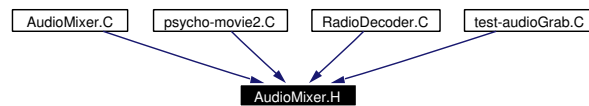
## 11.8 AudioMixer.H File Reference

```
#include "ModelComponent.H"
```

Include dependency graph for AudioMixer.H:



This graph shows which files directly or indirectly include this file:



### Classes

- class [AudioMixer](#)  
*Simple interface to the Linux audio mixer.*

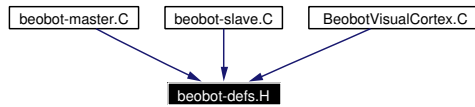
### 11.8.1 Detailed Description

A simple interface to select audio recording source

Definition in file [AudioMixer.H](#).

## 11.9 beobot-defs.H File Reference

This graph shows which files directly or indirectly include this file:



### Defines

- #define **IMGWIDTH** 160
- #define **IMGHEIGHT** 120
- #define **SMLEVEL** 2
- #define **LEVEL\_MIN** 0
- #define **LEVEL\_MAX** 2
- #define **DELTA\_MIN** 3
- #define **DELTA\_MAX** 4
- #define **NBORIENTS** 4
- #define **JETLEVEL** 3
- #define **JETDEPTH** 3
- #define **NBNEIGH** 24
- #define **NBCMAP** 8
- #define **NBJMAP** NBCMAP
- #define **BEO\_LUMFLICK** 1
- #define **BEO\_REDGREEN** 2
- #define **BEO\_BLUEYELLOW** 3
- #define **BEO\_ORI0\_45** 4
- #define **BEO\_ORI90\_135** 5
- #define **BEO\_CMAP** 6
- #define **BEO\_FMAP\_RG** 100
- #define **BEO\_FMAP\_BY** 101
- #define **BEO\_FMAP\_I** 102
- #define **BEO\_FMAP\_F** 103
- #define **BEO\_FMAP\_O0** 104
- #define **BEO\_FMAP\_O45** 105
- #define **BEO\_FMAP\_O90** 106
- #define **BEO\_FMAP\_O135** 107

### **11.9.1 Detailed Description**

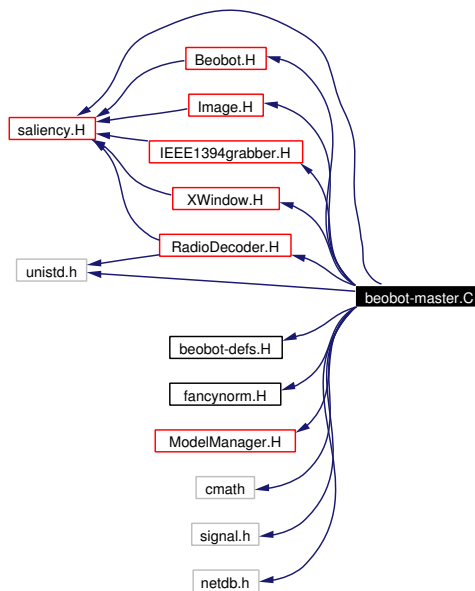
Basic definitions for the beobots

Definition in file [beobot-defs.H](#).

## 11.10 beobot-master.C File Reference

```
#include "saliency.H"  
#include "beobot-defs.H"  
#include "Beobot.H"  
#include "fancynorm.H"  
#include "Image.H"  
#include "IEEE1394grabber.H"  
#include "RadioDecoder.H"  
#include "XWindow.H"  
#include "ModelManager.H"  
#include <cmath>  
#include <unistd.h>  
#include <signal.h>  
#include <netdb.h>
```

Include dependency graph for beobot-master.C:



## Defines

- #define [SLAVES](#) "bb1ag:9790 bb1bg:9789 bb1bg:9790"  
*Names of the beowulf slaves (default port is 9789).*
- #define [NBAVG](#) 10  
*Number of frames over which to compute average stats:.*
- #define [NBSTATS](#) 6  
*Number of stats to collect:.*
- #define [USE\\_SINGLE\\_CPU](#) 0  
*define this to use only one cpu*

## Functions

- void [terminate](#) (int s)  
*Signal handler (e.g., for control-C).*
- int [main](#) (const int argc, const char \*\*argv)

## Variables

- bool [goforever](#) = true  
*Will turn false on interrupt signal.*

### 11.10.1 Detailed Description

main control program for beobots

Definition in file [beobot-master.C](#).

### 11.10.2 Define Documentation

#### 11.10.2.1 #define NBAVG 10

Number of frames over which to compute average stats:.

Definition at line 63 of file beobot-master.C.

Referenced by [main\(\)](#).

#### 11.10.2.2 **#define NBSTATS 6**

Number of stats to collect.

Definition at line 65 of file beobot-master.C.

Referenced by main().

#### 11.10.2.3 **#define SLAVES "bb1ag:9790 bb1bg:9789 bb1bg:9790"**

Names of the beowulf slaves (default port is 9789).

This parallel vision processing master is for use with beobot-slave.C. See the beobotgo script in bin/ for how to launch the slaves.

Definition at line 60 of file beobot-master.C.

Referenced by main().

#### 11.10.2.4 **#define USE\_SINGLE\_CPU 0**

define this to use only one cpu

Definition at line 68 of file beobot-master.C.

Referenced by main().

### 11.10.3 **Function Documentation**

#### 11.10.3.1 **int main (const int *argc*, const char \*\* *argv*)**

Definition at line 77 of file beobot-master.C.

References Image< PixRGB< byte > >::drawCircle(), XWindow::drawImage(), SharedPtr< T >::get(), Timer::getReset(), Beobot::getRetinaPtr(), BeobotAction::getSpeed(), BeobotAction::getTurn(), goforever, Point2D::i, Point2D::j, Beobot::lowLevel(), Beobot::lowLevelEnd(), Beobot::lowLevelStart(), main(), MYLOGVERB, NBAVG, NBSTATS, Timer::reset(), BeobotAction::setGear(), BeobotAction::setSpeed(), BeobotAction::setTurn(), SLAVES, terminate(), USE\_SINGLE\_CPU, and VCXNORM\_MAXNORM.

Referenced by main().

#### 11.10.3.2 **void terminate (int *s*)**

Signal handler (e.g., for control-C).

Definition at line 73 of file beobot-master.C.



References goforever.

Referenced by main().

#### 11.10.4 Variable Documentation

##### 11.10.4.1 `bool goforever = true` [static]

Will turn false on interrupt signal.

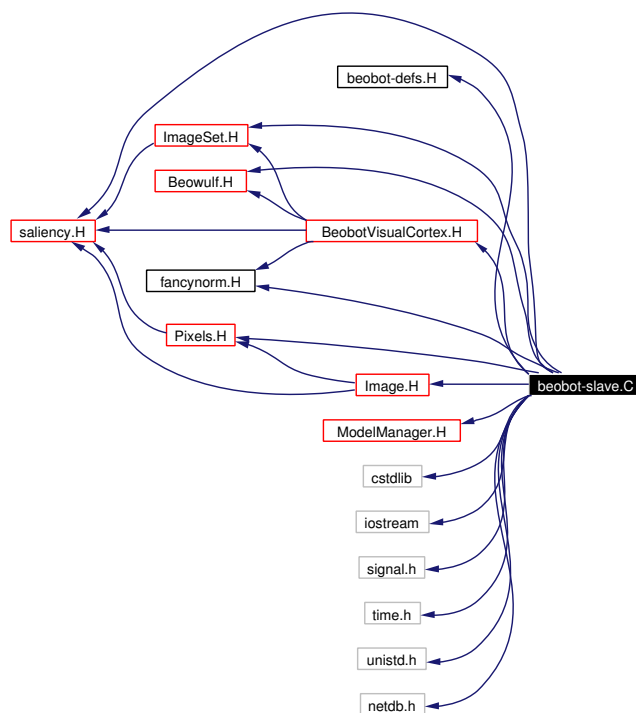
Definition at line 70 of file beobot-master.C.

Referenced by main(), and terminate().

## 11.11 beobot-slave.C File Reference

```
#include "saliency.H"  
#include "beobot-defs.H"  
#include "Beowulf.H"  
#include "fancynorm.H"  
#include "Pixels.H"  
#include "Image.H"  
#include "ImageSet.H"  
#include "BeobotVisualCortex.H"  
#include "ModelManager.H"  
#include <cstdlib>  
#include <iostream>  
#include <signal.h>  
#include <time.h>  
#include <unistd.h>  
#include <netdb.h>
```

Include dependency graph for beobot-slave.C:



## Defines

- `#define` **IMGWIDTH** 160
- `#define` **IMGHEIGHT** 120

## Functions

- void `terminate` (int s)  
*Signal handler (e.g., for control-C).*
- int `main` (const int argc, const char \*\*argv)

## Variables

- bool `goforever` = true  
*Will turn false on interrupt signal.*

### 11.11.1 Detailed Description

A parallel vision slave to use with beobotmaster. See the beobotgo script in bin/ for how to launch the slaves, and see beobot-master.C for the master program.

Definition in file [beobot-slave.C](#).

### 11.11.2 Function Documentation

#### 11.11.2.1 void terminate (int s)

Signal handler (e.g., for control-C).

Definition at line 64 of file beobot-slave.C.

References goforever.

### 11.11.3 Variable Documentation

#### 11.11.3.1 bool [goforever](#) = true [static]

Will turn false on interrupt signal.

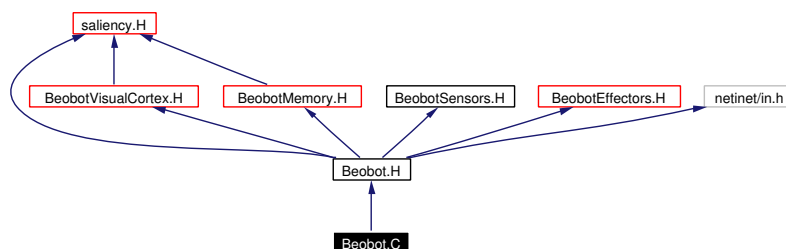
Definition at line 61 of file beobot-slave.C.

Referenced by terminate().

## 11.12 Beobot.C File Reference

```
#include "Beobot.H"
```

Include dependency graph for Beobot.C:



### 11.12.1 Detailed Description

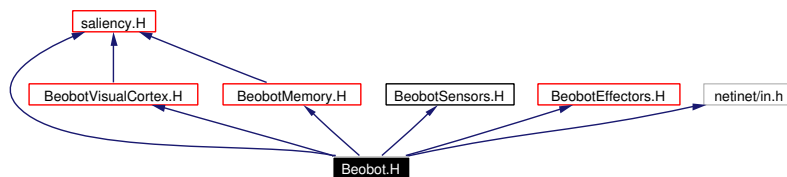
main [Beobot](#) class

Definition in file [Beobot.C](#).

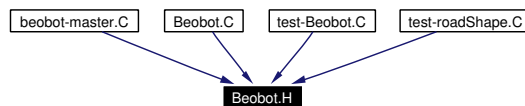
## 11.13 Beobot.H File Reference

```
#include "saliency.H"
#include "BeobotVisualCortex.H"
#include "BeobotMemory.H"
#include "BeobotSensors.H"
#include "BeobotEffectors.H"
#include <netinet/in.h>
```

Include dependency graph for Beobot.H:



This graph shows which files directly or indirectly include this file:



## Classes

- class [Beobot](#)  
*this class is a complete Beobot*

### 11.13.1 Detailed Description

Header file for the main [Beobot](#) class

Definition in file [Beobot.H](#).

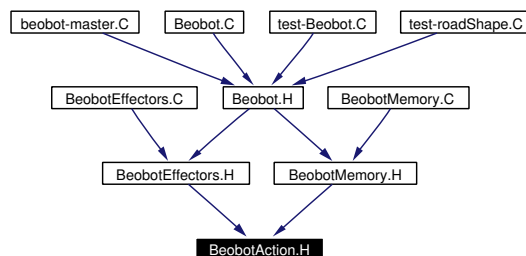
## 11.14 BeobotAction.H File Reference

```
#include "saliency.H"
```

```
#include "log.H"
```

Include dependency graph for BeobotAction.H:

This graph shows which files directly or indirectly include this file:



### Classes

- class [BeobotAction](#)

*This class defined motor actions for the Beobots.*

#### 11.14.1 Detailed Description

Motor actions for the Beobots

Definition in file [BeobotAction.H](#).

## 11.15 BeobotEffectors.C File Reference

```
#include "BeobotEffectors.H"
```

```
#include "lcd.H"
```

```
#include "ssc.H"
```

```
#include "carcontrol.H"
```

```
#include "ModelManager.H"
```

Include dependency graph for BeobotEffectors.C:

### 11.15.1 Detailed Description

controls the effectors of the Beobots (e.g., motors)

Definition in file [BeobotEffectors.C](#).

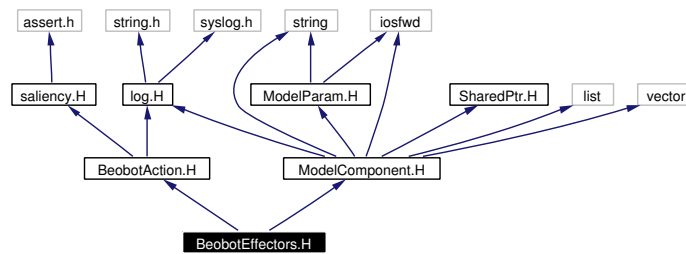


## 11.16 BeobotEffectors.H File Reference

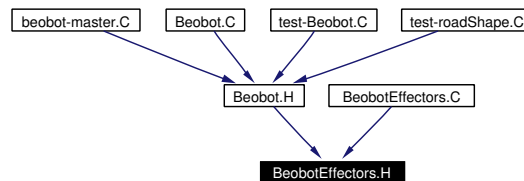
```
#include "BeobotAction.H"
```

```
#include "ModelComponent.H"
```

Include dependency graph for BeobotEffectors.H:



This graph shows which files directly or indirectly include this file:



### Classes

- class [BeobotEffectors](#)

*This class contains all the effectors of a [Beobot](#).*

### 11.16.1 Detailed Description

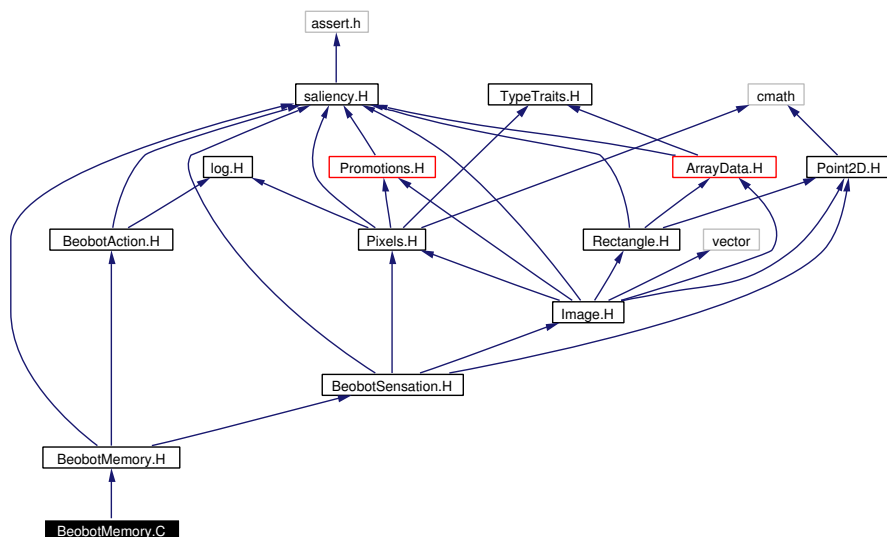
controls the effectors of the Beobots (e.g., motors)

Definition in file [BeobotEffectors.H](#).

## 11.17 BeobotMemory.C File Reference

```
#include "BeobotMemory.H"
```

Include dependency graph for BeobotMemory.C:



### 11.17.1 Detailed Description

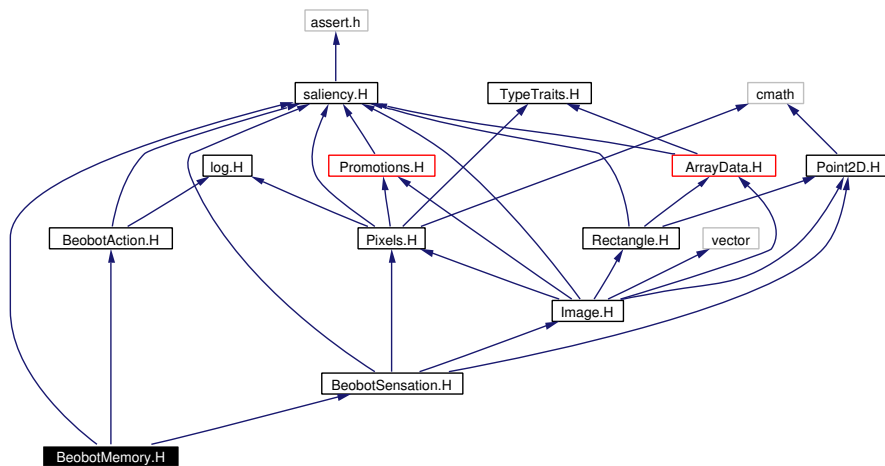
sensory/motor memory for the Beobots

Definition in file [BeobotMemory.C](#).

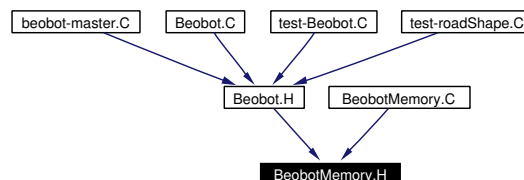
## 11.18 BeobotMemory.H File Reference

```
#include "saliency.H"
#include "BeobotSensation.H"
#include "BeobotAction.H"
```

Include dependency graph for BeobotMemory.H:



This graph shows which files directly or indirectly include this file:



## Classes

- class [BeobotMemory](#)  
*Memory of a [Beobot](#).*

## Enumerations

- enum **MemoryState** { HAVE\_SENSATION, HAVE\_RECOM\_ACTION, HAVE\_ACTION }

### 11.18.1 Detailed Description

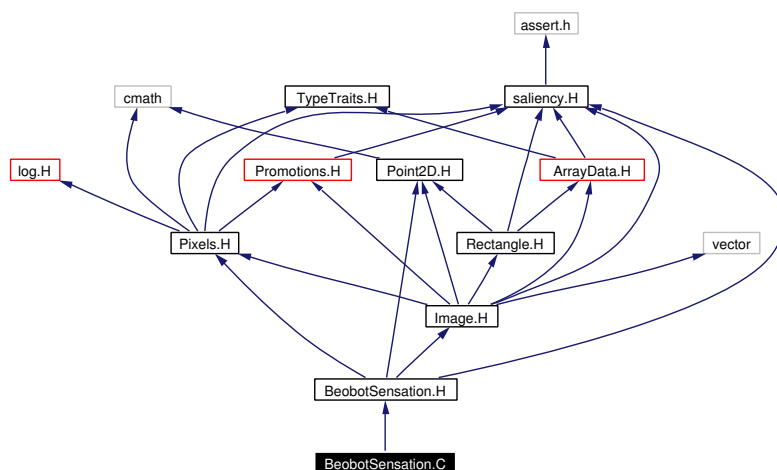
sensory/motor memory for the Beobots

Definition in file [BeobotMemory.H](#).

## 11.19 BeobotSensation.C File Reference

```
#include "BeobotSensation.H"
```

Include dependency graph for BeobotSensation.C:



### 11.19.1 Detailed Description

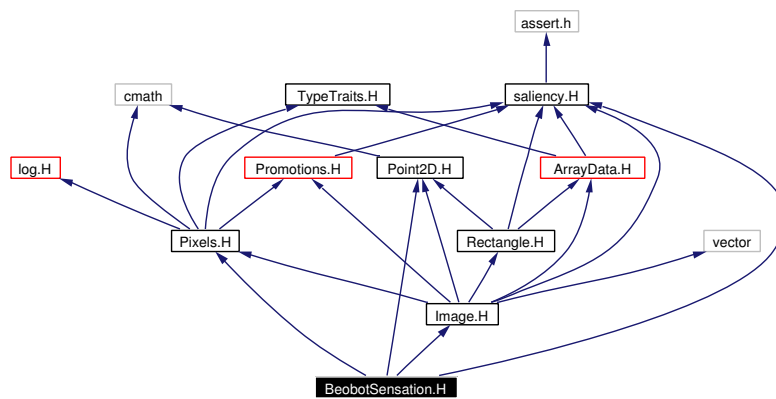
sensory events for the Beobots

Definition in file [BeobotSensation.C](#).

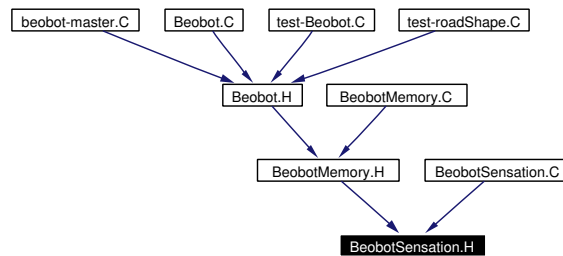
## 11.20 BeobotSensation.H File Reference

```
#include "saliency.H"
#include "Pixels.H"
#include "Image.H"
#include "Point2D.H"
```

Include dependency graph for BeobotSensation.H:



This graph shows which files directly or indirectly include this file:



## Classes

- class [BeobotSensation](#)

*this class contains all the sensory informations but not the action*

**11.20.1 Detailed Description**

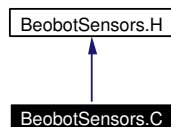
sensory events for the Beobots

Definition in file [BeobotSensation.H](#).

## 11.21 BeobotSensors.C File Reference

```
#include "BeobotSensors.H"
```

Include dependency graph for BeobotSensors.C:



### 11.21.1 Detailed Description

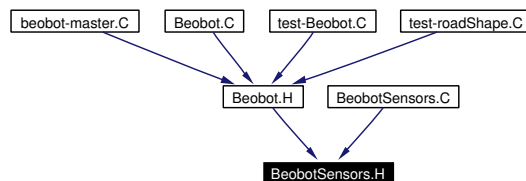
controls the sensors of the Beobots (e.g., speed)

Definition in file [BeobotSensors.C](#).



## 11.22 BeobotSensors.H File Reference

This graph shows which files directly or indirectly include this file:



### Classes

- class [BeobotSensors](#)  
*All the sensors of a [Beobot](#).*

### 11.22.1 Detailed Description

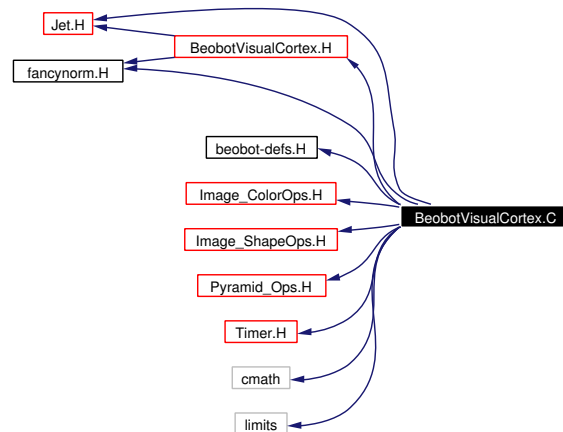
controls the sensors of the Beobots (e.g., speed)

Definition in file [BeobotSensors.H](#).

## 11.23 BeobotVisualCortex.C File Reference

```
#include "BeobotVisualCortex.H"  
#include "beobot-defs.H"  
#include "fancynorm.H"  
#include "Image_ColorOps.H"  
#include "Image_ShapeOps.H"  
#include "Jet.H"  
#include "Pyramid_Ops.H"  
#include "Timer.H"  
#include <cmath>  
#include <limits>
```

Include dependency graph for BeobotVisualCortex.C:



### 11.23.1 Detailed Description

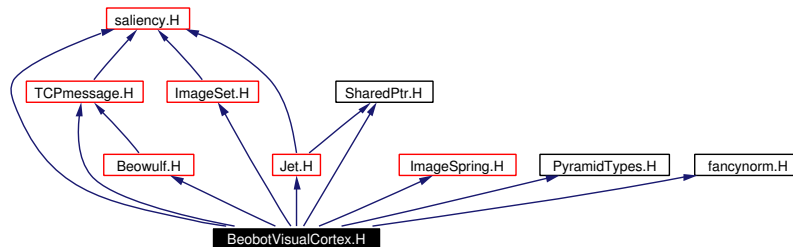
Implementation of navigation algorithm

Definition in file [BeobotVisualCortex.C](#).

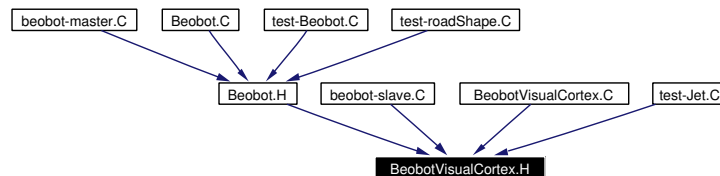
## 11.24 BeobotVisualCortex.H File Reference

```
#include "Beowulf.H"
#include "ImageSet.H"
#include "ImageSpring.H"
#include "Jet.H"
#include "PyramidTypes.H"
#include "saliency.H"
#include "SharedPtr.H"
#include "TCPmessage.H"
#include "fancynorm.H"
```

Include dependency graph for BeobotVisualCortex.H:



This graph shows which files directly or indirectly include this file:



## Classes

- class [BeobotVisualCortex](#)  
*Visual Cortex of a [Beobot](#).*

### 11.24.1 Detailed Description

Header for navigation class

Definition in file [BeobotVisualCortex.H](#).

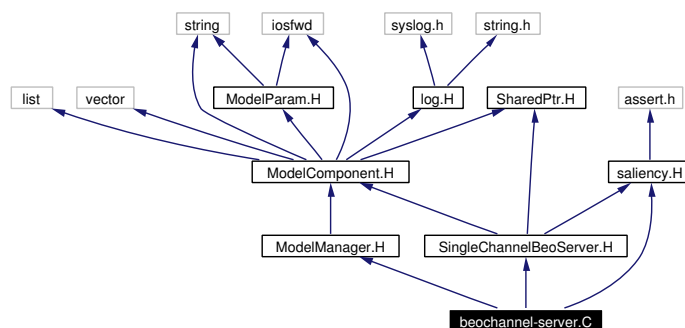
## 11.25 beochannel-server.C File Reference

```
#include "saliency.H"
```

```
#include "SingleChannelBeoServer.H"
```

```
#include "ModelManager.H"
```

Include dependency graph for beochannel-server.C:



### Functions

- int **main** (const int argc, const char \*\*argv)

#### 11.25.1 Detailed Description

a Beowulf-based [SingleChannel](#) server

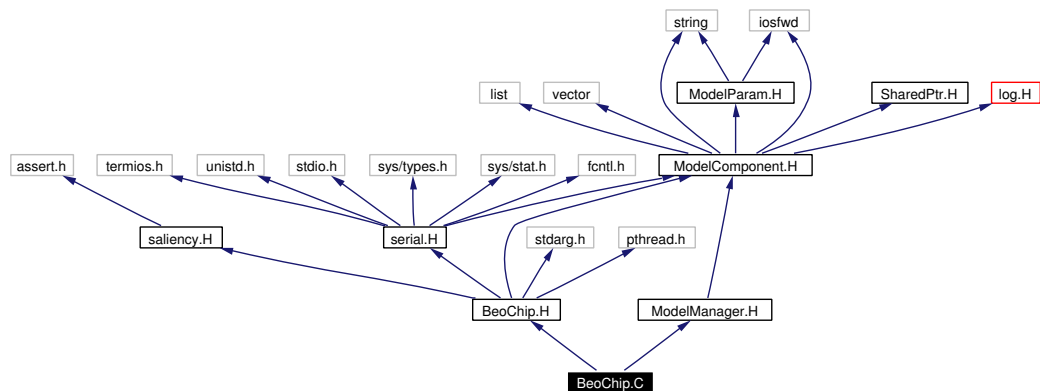
Definition in file [beochannel-server.C](#).

## 11.26 BeoChip.C File Reference

```
#include "BeoChip.H"
```

```
#include "ModelManager.H"
```

Include dependency graph for BeoChip.C:



### Classes

- class **binary**
- class **binary**< 0 >

### Defines

- #define **NUMSERVO** 8
- #define **NUMPWM** 2
- #define **NUMADC** 2
- #define **BIN**(N) (((byte)(binary<(unsigned long int)(N##.0)>::value))))

### Functions

- void \* **BeoChip\_run** (void \*c)

#### 11.26.1 Detailed Description

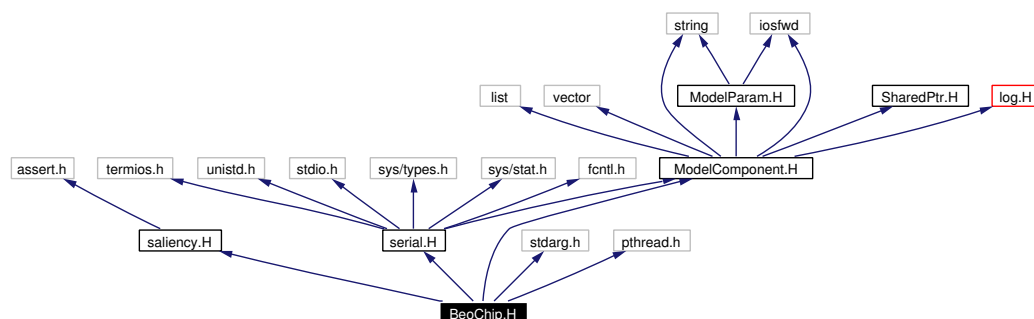
Interface to Brian Hudson's [BeoChip](#) interface device

Definition in file [BeoChip.C](#).

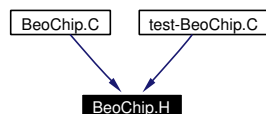
## 11.27 BeoChip.H File Reference

```
#include "saliency.H"
#include "ModelComponent.H"
#include "serial.H"
#include <stdarg.h>
#include <pthread.h>
```

Include dependency graph for BeoChip.H:



This graph shows which files directly or indirectly include this file:



## Classes

- class [BeoChipListener](#)  
*BeoChip event listener.*
- class [BeoChip](#)  
*BeoChip.H Interface to Brian Hudson's BeoChip device.*

## Enumerations

- enum [BeoChipEventType](#) {  
    **NONE** = 0, **PWM0** = 1, **PWM1** = 2, **KBD** = 3,  
    **ADC0** = 4, **ADC1** = 5, **RESET** = 6, **ECHOREP** = 7,  
    **INOVERFLOW** = 8, **SERIALERROR** = 9, **OUTOVERFLOW** = 10 }  
    *[BeoChip](#) event types.*

### 11.27.1 Detailed Description

Interface to Brian Hudson's [BeoChip](#) interface device.

Definition in file [BeoChip.H](#).

### 11.27.2 Enumeration Type Documentation

#### 11.27.2.1 enum [BeoChipEventType](#)

[BeoChip](#) event types.

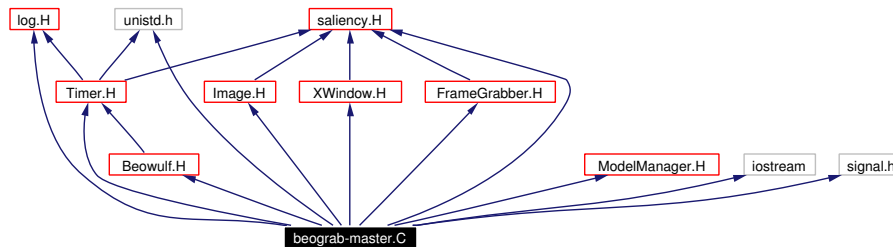
Definition at line 47 of file [BeoChip.H](#).



## 11.28 beograb-master.C File Reference

```
#include "saliency.H"
#include "log.H"
#include "Beowulf.H"
#include "Image.H"
#include "XWindow.H"
#include "FrameGrabber.H"
#include "Timer.H"
#include "ModelManager.H"
#include <iostream>
#include <unistd.h>
#include <signal.h>
```

Include dependency graph for beograb-master.C:



### Functions

- void **terminate** (int s)
- int **main** (const int argc, const char \*\*argv)

### Variables

- bool **goforever** = true

#### 11.28.1 Detailed Description

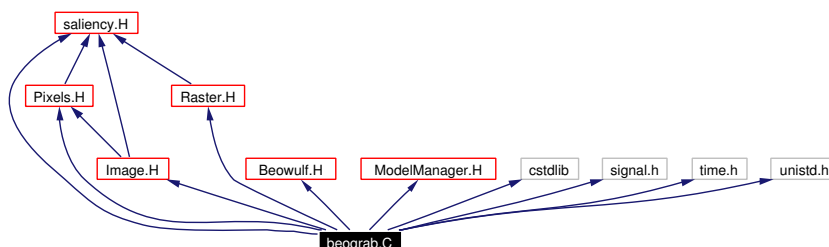
Grab video & save PNM frames on beowulf local disks

Definition in file [beograb-master.C](#).

## 11.29 beograb.C File Reference

```
#include "saliency.H"  
#include "Beowulf.H"  
#include "Pixels.H"  
#include "Image.H"  
#include "Raster.H"  
#include "ModelManager.H"  
#include <cstdlib>  
#include <signal.h>  
#include <time.h>  
#include <unistd.h>
```

Include dependency graph for beograb.C:



### Defines

- `#define OUTPATH "/home/tmp/1"`

### Functions

- void **terminate** (int s)
- int **main** (const int argc, const char \*\*argv)

### Variables

- bool **goforever** = true

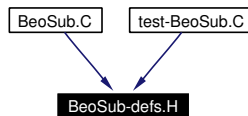
### 11.29.1 Detailed Description

Grab video & save PNM frames on beowulf local disks (slave)

Definition in file [beograb.C](#).

## 11.30 BeoSub-defs.H File Reference

This graph shows which files directly or indirectly include this file:



### Defines

- #define **BEOSUB\_SERDEV\_HMR3300** `"/dev/ttyUSB0"`  
*Serial port for the HMR3300 compass:.*
- #define **BEOSUB\_SERDEV\_DEPTH** `"/dev/ttyS0"`  
*Serial port for the pressure sensor:.*
- #define **BEOSUB\_SERDEV\_SSC** `"/dev/ttyS1"`  
*Serial port for the drop gate's SSC:.*
- #define **BEOSUB\_PARPORT\_ADDR** `0x378`  
*Parallel port address to use for the motor commands.*
- #define **BEOSUB\_GRAB\_W** `160`  
*Default grab dims.*
- #define **BEOSUB\_GRAB\_H** `120`
- #define **BEOSUB\_CAM\_FRONT** `1`  
*Camera numbers.*
- #define **BEOSUB\_CAM\_DOWN** `0`
- #define **BEOSUB\_CAM\_UP** `2`
- #define **GATEPASSDEPTH** `20`  
*Beosub Hardcoding Params.*
- #define **GATEPASSDIST** `40`
- #define **LIGHTSEEKDEPTH** `5`
- #define **LIGHTSEEKDIST** `30`
- #define **RECOVERTURN** `30`
- #define **RECOVERDIST** `60`
- #define **GATEDEPTH** `10.0f`

*Beosub mission params.*

- #define **GATEADVANCE** 100
- #define **LIGHTDEPTH** 15.0f
- #define **LIGHTDEPTHINCR** 0.5f
- #define **LIGHTADVANCE** 20
- #define **LIGHTPIXTHRESH** 50
- #define **VERBOSE** 0

*canny definitions*

- #define **BOOSTBLURFACTOR** 90.0
- #define **NUMFRAMES** 564
- #define **PATH** "/samples/1/frame"
- #define **NOEDGE** 0
- #define **POSSIBLE\_EDGE** 128
- #define **EDGE** 255
- #define **TOL** 2.0e-4
- #define **ITMAX** 200
- #define **SQR**(a) ((sqrarg=(a)) == 0.0 ? 0.0 : sqrarg\*sqrarg)
- #define **ITMAXB** 100
- #define **CGOLD** 0.3819660
- #define **ZEPS** 1.0e-10
- #define **SHFT**(a, b, c, d) (a)=(b);(b)=(c);(c)=(d);
- #define **SIGN**(a, b) ((b)>=0.0?fabs(a):-fabs(a))
- #define **GOLD** 1.618034
- #define **GLIMIT** 100.0
- #define **TINY** 1.0e-20
- #define **SHFT**(a, b, c, d) (a)=(b);(b)=(c);(c)=(d);
- #define **FMAX**(a, b)

### 11.30.1 Detailed Description

Defaults for the [BeoSub](#)

Definition in file [BeoSub-defs.H](#).

### 11.30.2 Define Documentation

#### 11.30.2.1 #define BEOSUB\_CAM\_FRONT 1

Camera numbers.

Definition at line 57 of file [BeoSub-defs.H](#).

Referenced by [main\(\)](#).

**11.30.2.2 #define BEOSUB\_GRAB\_W 160**

Default grab dims.

Definition at line 53 of file BeoSub-defs.H.

**11.30.2.3 #define BEOSUB\_PARPORT\_ADDR 0x378**

Parallel port address to use for the motor commands.

Definition at line 50 of file BeoSub-defs.H.

Referenced by BeoSub::BeoSub().

**11.30.2.4 #define BEOSUB\_SERDEV\_DEPTH "/dev/ttyS0"**

[Serial](#) port for the pressure sensor:.

Definition at line 44 of file BeoSub-defs.H.

Referenced by BeoSub::BeoSub().

**11.30.2.5 #define BEOSUB\_SERDEV\_HMR3300 "/dev/ttyUSB0"**

[Serial](#) port for the [HMR3300](#) compass:.

Definition at line 41 of file BeoSub-defs.H.

Referenced by BeoSub::BeoSub().

**11.30.2.6 #define BEOSUB\_SERDEV\_SSC "/dev/ttyS1"**

[Serial](#) port for the drop gate's [SSC](#):.

Definition at line 47 of file BeoSub-defs.H.

Referenced by BeoSub::BeoSub().

**11.30.2.7 #define FMAX(a, b)**

**Value:**

```
(maxarg1=(a), maxarg2=(b), (maxarg1)>(maxarg2)?\  
(maxarg1):(maxarg2))
```

Definition at line 99 of file BeoSub-defs.H.

**11.30.2.8 #define GATEDEPTH 10.0f**

Beosub mission params.

Definition at line 70 of file BeoSub-defs.H.

Referenced by main().

**11.30.2.9 #define GATEPASSDEPTH 20**

Beosub Hardcoding Params.

Definition at line 62 of file BeoSub-defs.H.

Referenced by main().

**11.30.2.10 #define VERBOSE 0**

canny definitions

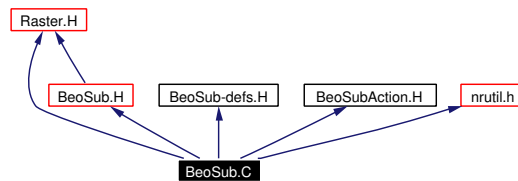
Definition at line 78 of file BeoSub-defs.H.



## 11.31 BeoSub.C File Reference

```
#include "BeoSub.H"
#include "BeoSub-defs.H"
#include "BeoSubAction.H"
#include "nrutil.h"
#include "Raster.H"
```

Include dependency graph for BeoSub.C:



### Functions

- int **floatcompare** (const void \*a, const void \*b)
- void **mnbrak** (double \*ax, double \*bx, double \*cx, double \*fa, double \*fb, double \*fc, double(\*func)(double))
- double **brent** (double ax, double bx, double cx, double(\*f)(double), double tol, double \*xmin)
- double **f1dim** (double x)
- void **linmin** (double p[ ], double xi[ ], int n, double \*fret, double(\*func)(double[ ]))
- void **assignDistMap** (Image< float > \*img)
- double **calcDistWrap** (double p[ ])
- float **getDistVal** (const float x, const float y)
- float **calcDist** (int x, int y, int w, int h, float alpha)

### Variables

- double **sqrarg**
- double **maxarg1**
- double **maxarg2**
- int **ncom**
- double \* **pcom**
- double \* **xicom**

- `double(* nrfunc )(double[ ])`
- `SharedPtr< XWindow > beoSubXwin`
- `const bool debugmode = true`
- `Image< PixRGB< byte > > beoSubXdisp`
- `Image< float > beoSubDistMap`

### 11.31.1 Detailed Description

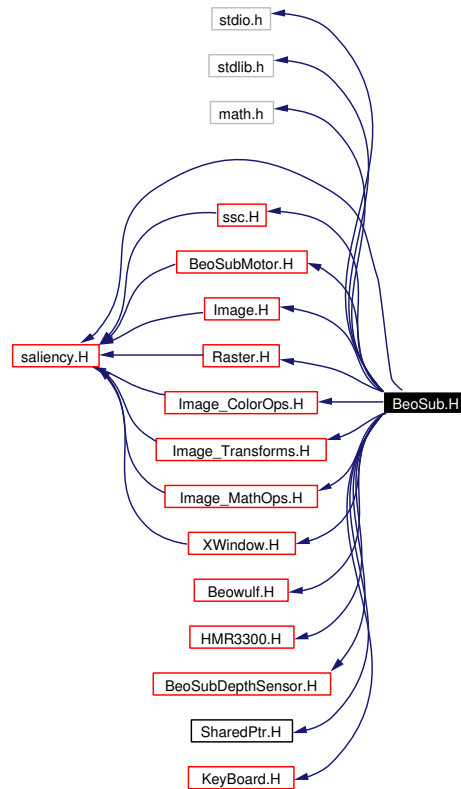
An autonomous submarine

Definition in file [BeoSub.C](#).

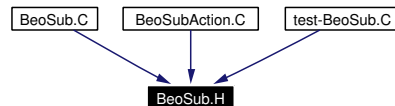
## 11.32 BeoSub.H File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "saliency.H"
#include "Beowulf.H"
#include "HMR3300.H"
#include "ssc.H"
#include "BeoSubMotor.H"
#include "BeoSubDepthSensor.H"
#include "Image.H"
#include "Raster.H"
#include "Image_ColorOps.H"
#include "Image_Transforms.H"
#include "Image_MathOps.H"
#include "XWindow.H"
#include "SharedPtr.H"
#include "KeyBoard.H"
```

Include dependency graph for BeoSub.H:



This graph shows which files directly or indirectly include this file:



## Classes

- class [BeoSub](#)

*Definition and access functions for the BeoSub.*

## Functions

- double **calcDistWrap** (double p[ ])

- float **getDistVal** (const float x, const float y)
- float **calcDist** (int x, int y, int w, int h, float alpha)
- void **assignDistMap** ([Image](#)< float > \*img)

### 11.32.1 Detailed Description

An autonomous submarine

Definition in file [BeoSub.H](#).

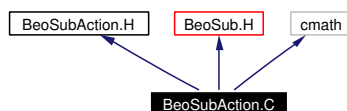
## 11.33 BeoSubAction.C File Reference

```
#include "BeoSubAction.H"
```

```
#include "BeoSub.H"
```

```
#include <cmath>
```

Include dependency graph for BeoSubAction.C:



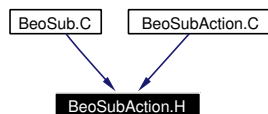
### 11.33.1 Detailed Description

Helper class for [BeoSub](#) motor actions

Definition in file [BeoSubAction.C](#).

## 11.34 BeoSubAction.H File Reference

This graph shows which files directly or indirectly include this file:



### Classes

- class [BeoSubAction](#)  
*Helper base class for [BeoSub](#) motor actions.*
- class [BeoSubActionDive](#)  
*Helper base class for [BeoSub](#) dive actions.*
- class [BeoSubActionTurn](#)  
*Helper base class for [BeoSub](#) turn actions.*

### 11.34.1 Detailed Description

Helper class for [BeoSub](#) motor actions

Definition in file [BeoSubAction.H](#).

## 11.35 BeoSubDepthSensor.C File Reference

```
#include "BeoSubDepthSensor.H"
```

```
#include <sstream>
```

```
#include <string>
```

```
#include <cstdio>
```

```
#include <unistd.h>
```

Include dependency graph for BeoSubDepthSensor.C:

### Functions

- void \* **Sensor\_run** (void \*c)

#### 11.35.1 Detailed Description

class for interfacing with a depth sensor

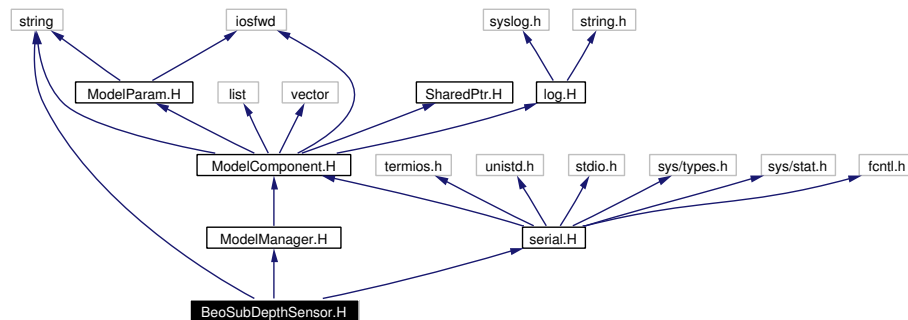
Definition in file [BeoSubDepthSensor.C](#).



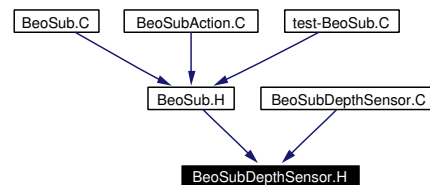
## 11.36 BeoSubDepthSensor.H File Reference

```
#include <string>
#include "ModelManager.H"
#include "serial.H"
```

Include dependency graph for BeoSubDepthSensor.H:



This graph shows which files directly or indirectly include this file:



### Classes

- class [BeoSubDepthSensor](#)  
*Class for interfacing with a depth sensor.*

#### 11.36.1 Detailed Description

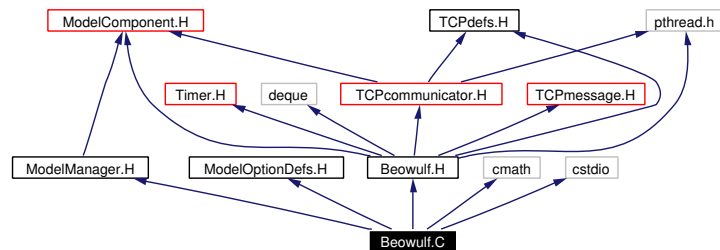
class for interfacing with a depth sensor

Definition in file [BeoSubDepthSensor.H](#).

## 11.37 Beowulf.C File Reference

```
#include "Beowulf.H"  
#include "ModelOptionDefs.H"  
#include "ModelManager.H"  
#include <cmath>  
#include <stdio>
```

Include dependency graph for Beowulf.C:



### 11.37.1 Detailed Description

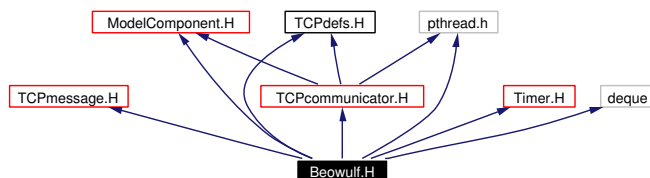
Simple interfacing to a [Beowulf](#) cluster

Definition in file [Beowulf.C](#).

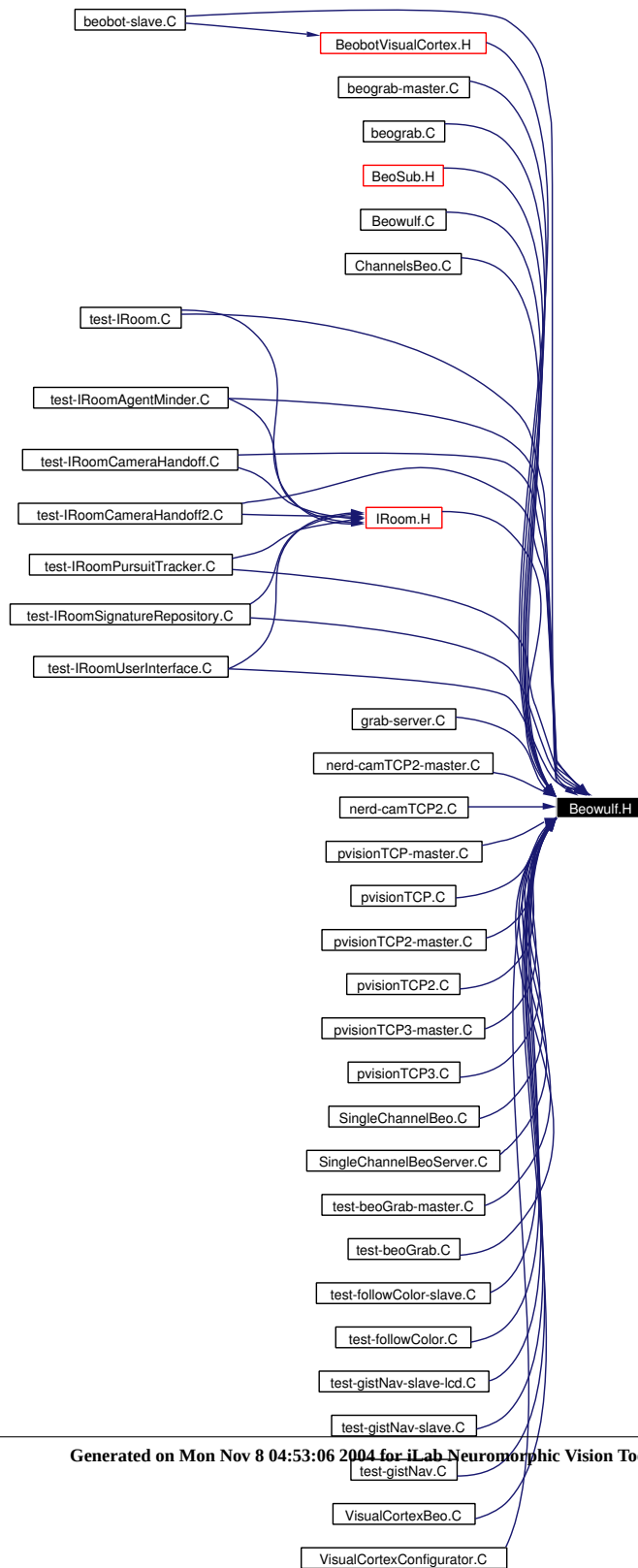
## 11.38 Beowulf.H File Reference

```
#include "TCPmessage.H"  
#include "ModelComponent.H"  
#include "TCPcommunicator.H"  
#include "TCPdefs.H"  
#include "Timer.H"  
#include <deque>  
#include <pthread.h>
```

Include dependency graph for Beowulf.H:



This graph shows which files directly or indirectly include this file:



## Classes

- class [Beowulf](#)  
*Simple interfacing to a Beowulf cluster.*

## Defines

- #define **BEO\_NONE** 0
- #define **BEO\_INIT** -1
- #define **BEO\_INIT2** -2
- #define **BEO\_INIT3** -3

### 11.38.1 Detailed Description

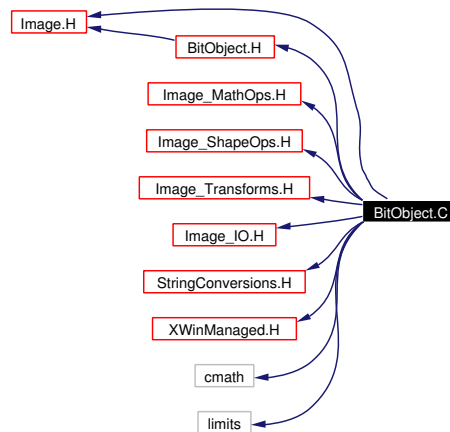
Simple interfacing to a [Beowulf](#) cluster

Definition in file [Beowulf.H](#).

## 11.39 BitObject.C File Reference

```
#include "BitObject.H"
#include "Image.H"
#include "Image_MathOps.H"
#include "Image_ShapeOps.H"
#include "Image_Transforms.H"
#include "Image_IO.H"
#include "StringConversions.H"
#include "XWinManaged.H"
#include <cmath>
#include <limits>
```

Include dependency graph for BitObject.C:



### Defines

- #define **INSTITUTE**(T\_or\_RGB)

### Functions

- **INSTITUTE** (PixRGB< byte >)
- **INSTITUTE** (PixRGB< float >)
- **INSTITUTE** (byte)

- **INstantiate** (float)

### 11.39.1 Detailed Description

class that holds an object defined by a shape in a bit image

Definition in file [BitObject.C](#).

### 11.39.2 Define Documentation

#### 11.39.2.1 #define INstantiate(T\_or\_RGB)

**Value:**

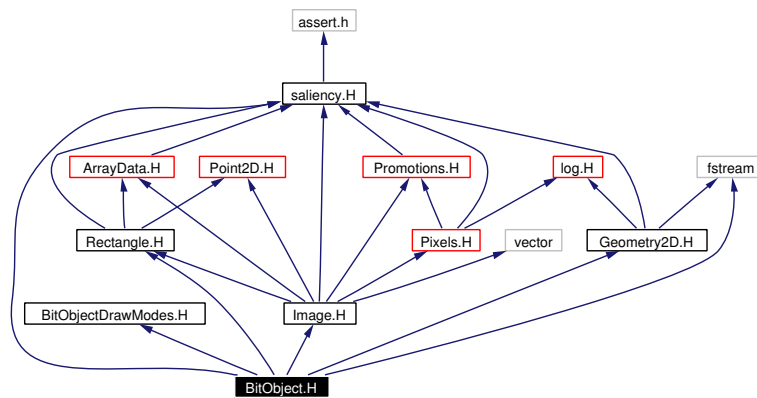
```
template void BitObject::drawShape(Image< T_or_RGB >& img, \
                                   const T_or_RGB& color, \
                                   float opacity); \
template void BitObject::drawOutline(Image< T_or_RGB >& img, \
                                     const T_or_RGB& color, \
                                     float opacity); \
template void BitObject::drawBoundingBox(Image< T_or_RGB >& img, \
                                         const T_or_RGB& color, \
                                         float opacity); \
template void BitObject::draw(BitObjectDrawMode mode, \
                              Image< T_or_RGB >& img, \
                              const T_or_RGB& color, \
                              float opacity);
```

Definition at line 706 of file BitObject.C.

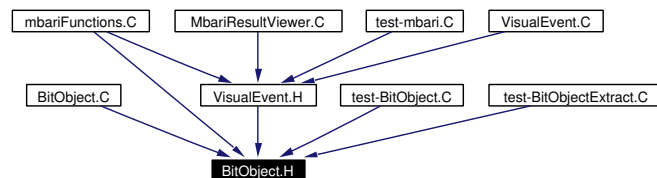
## 11.40 BitObject.H File Reference

```
#include "saliency.H"
#include "BitObjectDrawModes.H"
#include "Rectangle.H"
#include "Image.H"
#include "Geometry2D.H"
#include <fstream>
```

Include dependency graph for BitObject.H:



This graph shows which files directly or indirectly include this file:



## Classes

- class [BitObject](#)

*Object defined by a connected binary pixel region.*



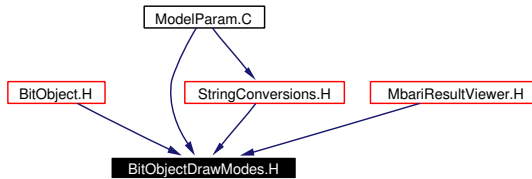
### 11.40.1 Detailed Description

class that holds an object defined by a shape in a bit image

Definition in file [BitObject.H](#).

## 11.41 BitObjectDrawModes.H File Reference

This graph shows which files directly or indirectly include this file:



### Defines

- `#define NBBITOBJECTDRAWMODES 4`  
*number of known [BitObject](#) draw modes:*

### Enumerations

- `enum BitObjectDrawMode { BODMnone = 0, BODMshape = 1, BODMoutline = 2, BODMbbox = 3 }`  
*Possible [BitObject](#) draw modes.*

### Functions

- `const char * bitObjectDrawModeName (const BitObjectDrawMode p)`  
*Returns name of given [BitObject](#) draw mode.*

#### 11.41.1 Detailed Description

definition of the different modes of drawing a [BitObject](#) into a frame

Definition in file [BitObjectDrawModes.H](#).

#### 11.41.2 Define Documentation

##### 11.41.2.1 `#define NBBITOBJECTDRAWMODES 4`

number of known [BitObject](#) draw modes:

Definition at line 51 of file BitObjectDrawModes.H.

Referenced by bitObjectDrawModeName(), and convertFromString().

### 11.41.3 Enumeration Type Documentation

#### 11.41.3.1 enum BitObjectDrawMode

Possible BitObject draw modes.

**Enumeration values:**

*BODMshape* don't draw the BitObject

*BODMoutline* draw the shape of the BitObject

*BODMbbox* draw the outline of the BitObject

Definition at line 42 of file BitObjectDrawModes.H.

Referenced by convertFromString().

#### 11.41.4 Function Documentation

##### 11.41.4.1 const char\* bitObjectDrawModeName (const BitObjectDrawMode p) [inline]

Returns name of given BitObject draw mode.

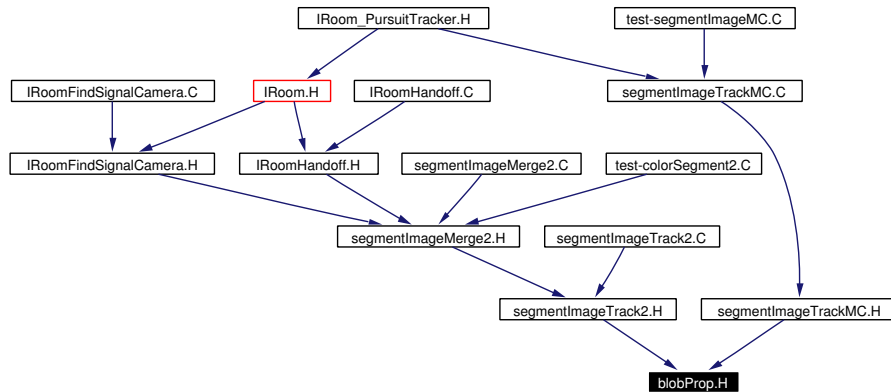
Definition at line 54 of file BitObjectDrawModes.H.

References NBBITOBJECTDRAWMODES.

Referenced by convertFromString(), and convertToString().

## 11.42 blobProp.H File Reference

This graph shows which files directly or indirectly include this file:



### Classes

- class **blobProp**

### 11.42.1 Detailed Description

Basic image segmenter blob finder using color part of color tracker

Definition in file [blobProp.H](#).

## 11.43 BlurFoveator.C File Reference

```
#include "BlurFoveator.H"
```

```
#include <cmath>
```

Include dependency graph for BlurFoveator.C:

### 11.43.1 Detailed Description

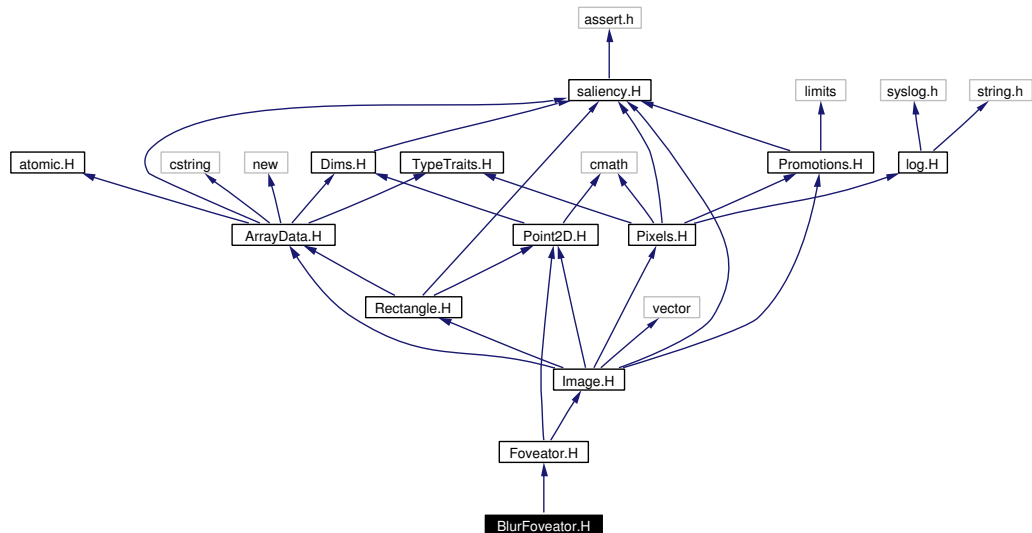
[Foveator](#) class that performs progressive blurring

Definition in file [BlurFoveator.C](#).

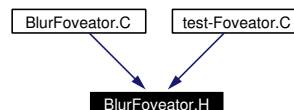
## 11.44 BlurFoveator.H File Reference

```
#include "Foveator.H"
```

Include dependency graph for BlurFoveator.H:



This graph shows which files directly or indirectly include this file:



## Classes

- class [BlurFoveator](#)

*A class that does space-variant processing with progressive blurring.*

- class **BlurFoveator::GaussianBlurMatrix**

*An inner class that defines a gaussian blur matrix.*

### 11.44.1 Detailed Description

[Foveator](#) class that performs progressive blurring

Definition in file [BlurFoveator.H](#).

## 11.45 bmcvfigs.C File Reference

```
#include "Image_All.H"
#include "Image_DrawOps.H"
#include "log.H"
#include "Raster.H"
#include "Brain.H"
#include "VisualCortex.H"
#include "SaccadeControllers.H"
#include "ChannelBase.H"
#include "SharedPtr.H"
#include "fancynorm.H"
#include <iostream>
#include <stdio.h>
#include "SingleChannel.H"
#include "ModelManager.H"
#include "StdBrain.H"
#include "StringConversions.H"
#include "ShapeEstimator.H"
#include "ModelOptionDefs.H"
```

Include dependency graph for bmcvfigs.C:

### Defines

- #define [MAX\\_TIME\\_DEFAULT](#) 0.7  
*max time in seconds*

### Functions

- int [main](#) (const int argc, const char \*\*argv)  
*The main routine.*



### 11.45.1 Detailed Description

extract size information from the SM

Definition in file [bmcvfigs.C](#).

### 11.45.2 Define Documentation

#### 11.45.2.1 `#define MAX_TIME_DEFAULT 0.7`

max time in seconds

Definition at line 62 of file bmcvfigs.C.

Referenced by `main()`.

### 11.45.3 Function Documentation

#### 11.45.3.1 `int main (const int argc, const char ** argv)`

The main routine.

Definition at line 69 of file bmcvfigs.C.

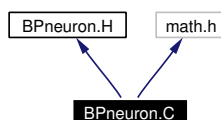
References `contrastModulate()`, `convertFromString()`, `FULLTRACE`, `Image< T >::getDims()`, `Point2D::i`, `Image< T >::lowThresh()`, `main()`, `MYLOGVERB`, `Image< T >::normalize()`, `OPTEXP_CORE`, `Raster::ReadRGB()`, `scaleBlock()`, `toStr()`, and `Raster::WriteRGB()`.

## 11.46 BPneuron.C File Reference

```
#include "BPneuron.H"
```

```
#include <math.h>
```

Include dependency graph for BPneuron.C:



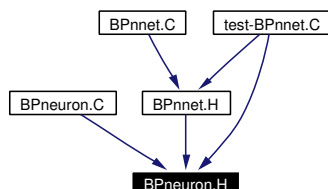
### 11.46.1 Detailed Description

Back Prop Neuron class

Definition in file [BPneuron.C](#).

## 11.47 BPneuron.H File Reference

This graph shows which files directly or indirectly include this file:



### Classes

- class [BPneuron](#)  
*neuron for a back prop neural net ([BPnnnet](#))*

#### 11.47.1 Detailed Description

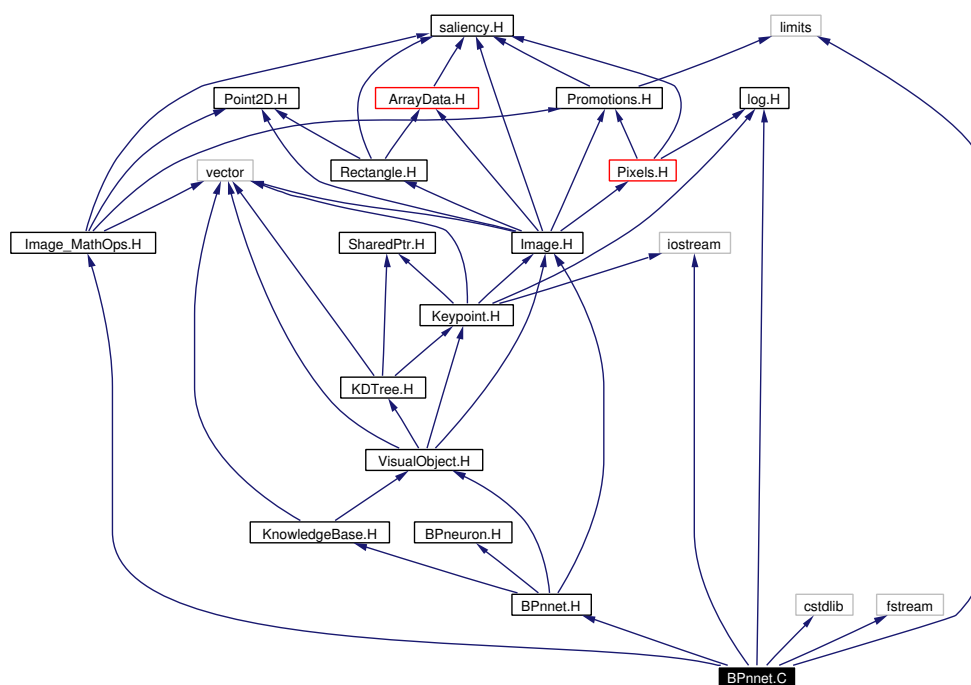
header for a back prop neuron

Definition in file [BPneuron.H](#).

## 11.48 BPnnnet.C File Reference

```
#include "BPnnnet.H"
#include <cstdlib>
#include <limits>
#include <fstream>
#include <iostream>
#include "Image_MathOps.H"
#include "log.H"
```

Include dependency graph for BPnnnet.C:



### 11.48.1 Detailed Description

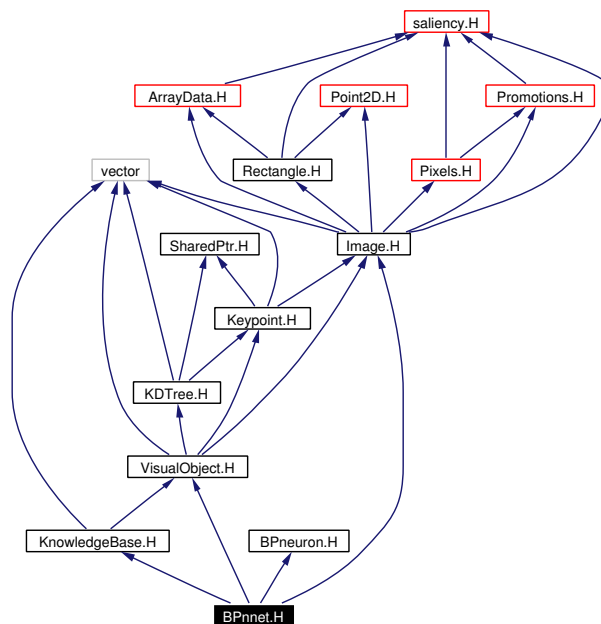
## Back Prop Neural Net class

Definition in file [BPnnnet.C](#).

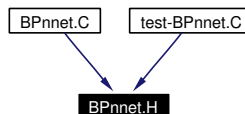
## 11.49 BPnnnet.H File Reference

```
#include "VisualObject.H"
#include "BPneuron.H"
#include "KnowledgeBase.H"
#include "Image.H"
```

Include dependency graph for BPnnnet.H:



This graph shows which files directly or indirectly include this file:



### Classes

- class [BPnnnet](#)  
*describes structure of a 3 layer back prop neural net*

### 11.49.1 Detailed Description

header for Back Prop Neural Net

Definition in file [BPnnnet.H](#).

## 11.50 Brain.C File Reference

```
#include "Brain.H"
#include "Image_ColorOps.H"
#include "Image_FilterOps.H"
#include "Image_MathOps.H"
#include "Image_ShapeOps.H"
#include "Image_Transforms.H"
#include "Pyramid_Ops.H"
#include "Retina.H"
#include "VisualCortex.H"
#include "VisualCortexEyeMvt.H"
#include "Point2DT.H"
#include "Channels.H"
#include "ShapeEstimator.H"
#include "SimulationViewer.H"
#include "ModelManager.H"
#include "SaccadeController.H"
#include "SaliencyMap.H"
#include "TaskRelevanceMap.H"
#include "AttentionGuidanceMap.H"
#include "WinnerTakeAll.H"
#include "Raster.H"
#include "FrameSeries.H"
#include "ModelOptionDefs.H"
#include "Keypoint.H"
#include "InferoTemporal.H"
#include "GistEstimator.H"
#include <cstdio>
```

Include dependency graph for Brain.C:

### 11.50.1 Detailed Description

The whole brain

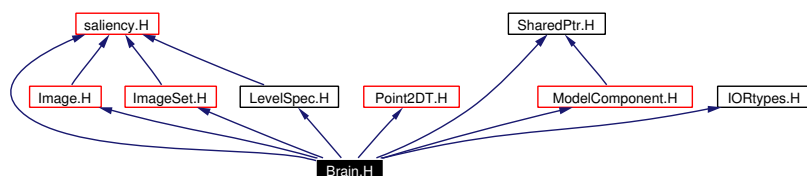
Definition in file [Brain.C](#).



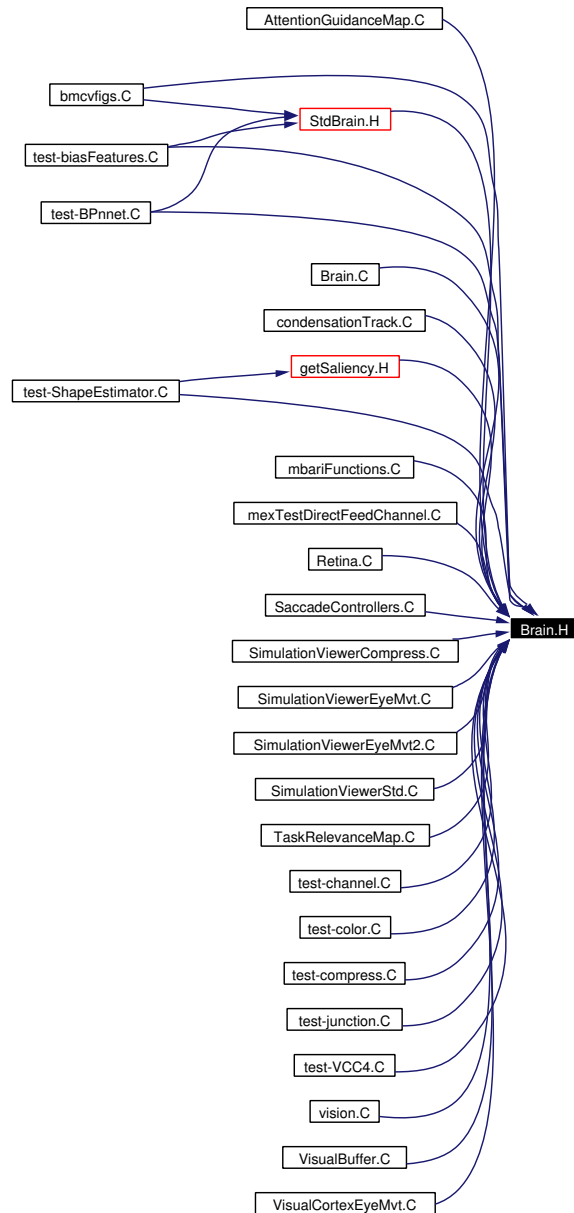
## 11.51 Brain.H File Reference

```
#include "saliency.H"  
#include "Point2DT.H"  
#include "Image.H"  
#include "SharedPtr.H"  
#include "ImageSet.H"  
#include "ModelComponent.H"  
#include "IORtypes.H"  
#include "LevelSpec.H"
```

Include dependency graph for Brain.H:



This graph shows which files directly or indirectly include this file:



## Classes

- class [Brain](#)

*This is the brain, a holder for various vision objects.*

**11.51.1 Detailed Description**

The whole brain

Definition in file [Brain.H](#).

## 11.52 `calibrateCamera.C` File Reference

```
#include "ssc.H"
#include "Image.H"
#include "IEEE1394grabber.H"
#include "XWindow.H"
#include "cameraConf.H"
#include "Pixels.H"
#include "ModelManager.H"
#include <stdio>
#include <stdlib>
#include <stdlib.h>
#include <iostream>
#include <signal.h>
#include <unistd.h>
Include dependency graph for calibrateCamera.C:
```

### Functions

- `int main` (`const int argc`, `const char **argv`)

#### 11.52.1 Detailed Description

FIXME

Definition in file [calibrateCamera.C](#).

## 11.53 `calibrateFunctions.C` File Reference

```
#include "calibrateFunctions.H"
```

```
#include "Image.H"
```

```
#include <vector>
```

```
#include <iostream>
```

Include dependency graph for `calibrateFunctions.C`:

### Functions

- double **getError** (const [Image](#)< [byte](#) > &img1, const [Image](#)< [byte](#) > &img2, const int shiftx, const int shifty)
- int **findAlign** (const [Image](#)< [byte](#) > &img1, const [Image](#)< [byte](#) > &img2, const int align\_xy, const int range\_start, const int range\_end)

### 11.53.1 Detailed Description

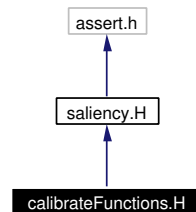
FIXME

Definition in file [calibrateFunctions.C](#).

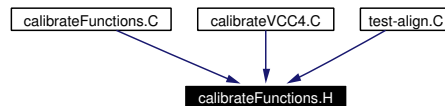
## 11.54 `calibrateFunctions.H` File Reference

```
#include "saliency.H"
```

Include dependency graph for `calibrateFunctions.H`:



This graph shows which files directly or indirectly include this file:



### Defines

- `#define ALIGN_X 0`
- `#define ALIGN_Y 1`

### Functions

- double **getError** (const `Image< byte >` &img1, const `Image< byte >` &img2, const int shiftx, const int shifty)
- int **findAlign** (const `Image< byte >` &img1, const `Image< byte >` &img2, const int align\_xy, const int range\_start, const int range\_end)

#### 11.54.1 Detailed Description

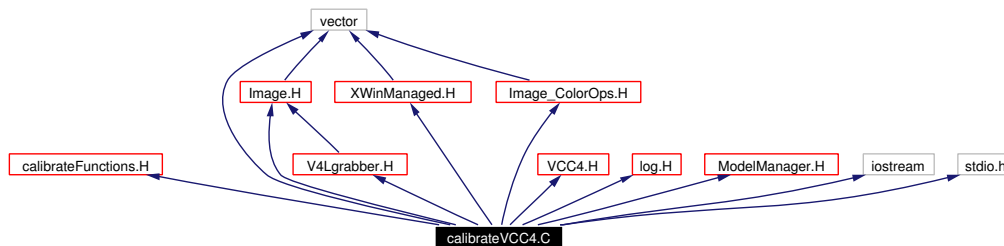
Helper functions for camera calibration

Definition in file [calibrateFunctions.H](#).

## 11.55 calibrateVCC4.C File Reference

```
#include "calibrateFunctions.H"
#include "V4Lgrabber.H"
#include "VCC4.H"
#include "XWinManaged.H"
#include "Image.H"
#include "Image_ColorOps.H"
#include "log.H"
#include "ModelManager.H"
#include <iostream>
#include <stdio.h>
#include <vector>
```

Include dependency graph for calibrateVCC4.C:



## Functions

- int **main** (const int argc, const char \*\*argv)

### 11.55.1 Detailed Description

Calibration for the [VCC4](#) pan/tilt camera unit

Definition in file [calibrateVCC4.C](#).

## 11.56 cameraConf.C File Reference

```
#include "cameraConf.H"
```

Include dependency graph for cameraConf.C:

### 11.56.1 Detailed Description

FIXME

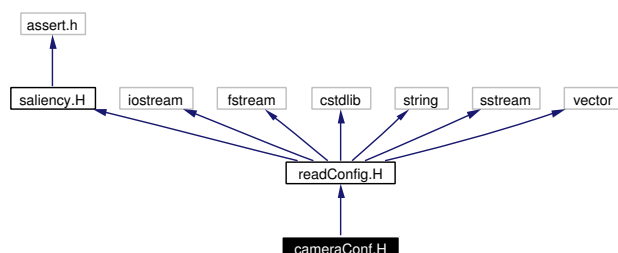
Definition in file [cameraConf.C](#).



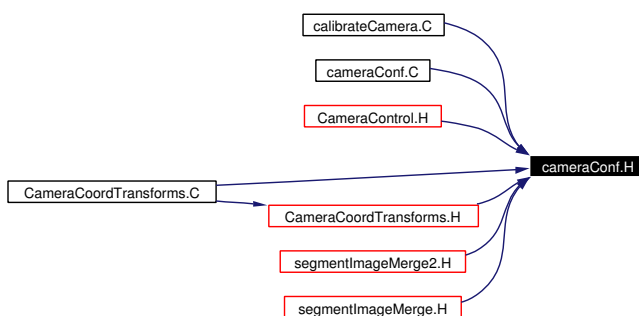
## 11.57 cameraConf.H File Reference

```
#include "readConfig.H"
```

Include dependency graph for cameraConf.H:



This graph shows which files directly or indirectly include this file:



### Classes

- class [cameraConf](#)

*This class uses [readConfig](#) to extract config parameters for [CameraControl](#).*

### 11.57.1 Detailed Description

A set of camera configuration parameters

Definition in file [cameraConf.H](#).

## 11.58 CameraControl.C File Reference

```
#include "CameraControl.H"
```

```
#include <math.h>
```

```
#include <unistd.h>
```

```
#include <signal.h>
```

```
#include "ssc.H"
```

Include dependency graph for CameraControl.C:

### 11.58.1 Detailed Description

Controls a pan/tilt camera head

Definition in file [CameraControl.C](#).

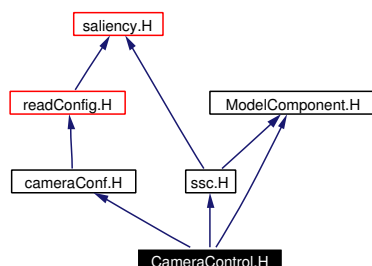
## 11.59 CameraControl.H File Reference

```
#include "cameraConf.H"
```

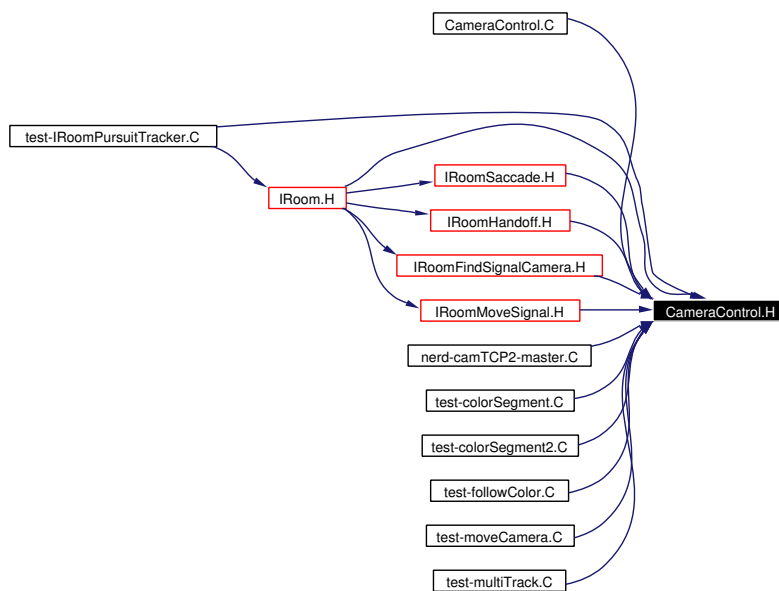
```
#include "ssc.H"
```

```
#include "ModelComponent.H"
```

Include dependency graph for CameraControl.H:



This graph shows which files directly or indirectly include this file:



## Classes

- class [CameraControl](#)

*This class is an interface to pan, tilt, and otherwise move the camera.*

### 11.59.1 Detailed Description

Controls a pan/tilt camera head

Definition in file [CameraControl.H](#).

## 11.60 carConf.C File Reference

```
#include "carConf.H"
```

Include dependency graph for carConf.C:

### 11.60.1 Detailed Description

A set of car configuration parameters

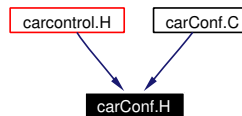
Definition in file [carConf.C](#).

## 11.61 carConf.H File Reference

```
#include "readConfig.H"
```

Include dependency graph for carConf.H:

This graph shows which files directly or indirectly include this file:



### Classes

- class [carConf](#)

*This class uses [readConfig](#) to extract config parameters for [CarControl](#).*

### 11.61.1 Detailed Description

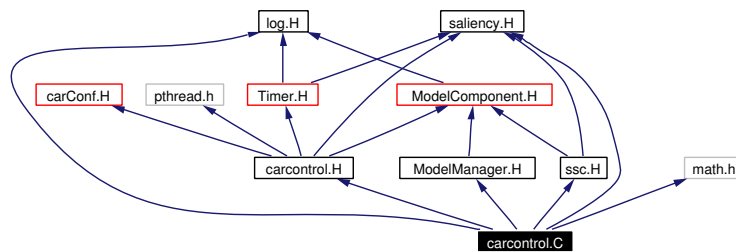
A set of car configuration parameters

Definition in file [carConf.H](#).

## 11.62 carcontrol.C File Reference

```
#include "carcontrol.H"  
#include "log.H"  
#include "ssc.H"  
#include "saliency.H"  
#include "ModelManager.H"  
#include <math.h>
```

Include dependency graph for carcontrol.C:



### Functions

- void \* **speedRampFunc** (void \*ptr)

#### 11.62.1 Detailed Description

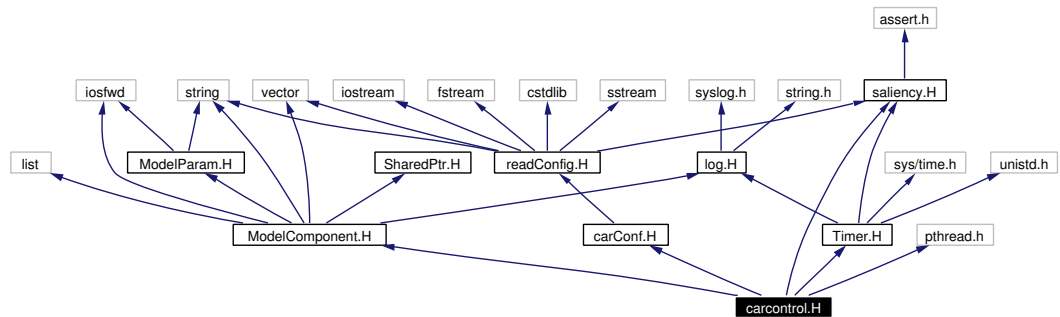
control the various servos in an R/C car

Definition in file [carcontrol.C](#).

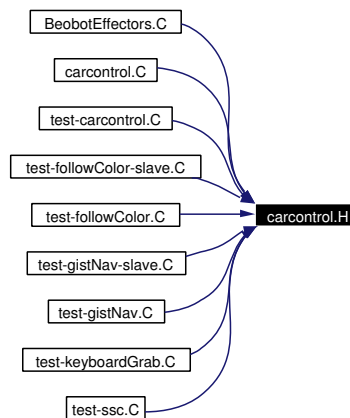
## 11.63 carcontrol.H File Reference

```
#include "ModelComponent.H"
#include "carConf.H"
#include "Timer.H"
#include "saliency.H"
#include <pthread.h>
```

Include dependency graph for carcontrol.H:



This graph shows which files directly or indirectly include this file:



## Classes

- class [CarControl](#)



*This class is an interface to pilot the RC car.*

## Enumerations

- enum [SpeedRampType](#) { [SPEED\\_RAMP\\_LINEAR](#) = 0, [SPEED\\_RAMP\\_SIGMOID](#) = 1 }

*Speed ramping options.*

### 11.63.1 Detailed Description

control the various servos in an R/C car

Definition in file [carcontrol.H](#).

### 11.63.2 Enumeration Type Documentation

#### 11.63.2.1 enum [SpeedRampType](#)

Speed ramping options.

**Enumeration values:**

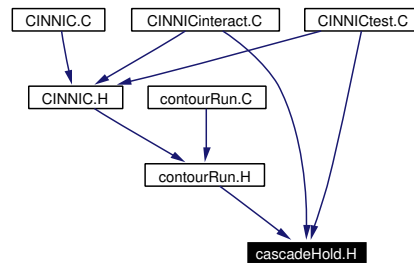
**[SPEED\\_RAMP\\_LINEAR](#)** use linear ramping

**[SPEED\\_RAMP\\_SIGMOID](#)** use sigmoid ramping

Definition at line 48 of file [carcontrol.H](#).

## 11.64 cascadeHold.H File Reference

This graph shows which files directly or indirectly include this file:



### Classes

- class [cascadeHold](#)

*This is a container class for holding cascading energy values.*

### 11.64.1 Detailed Description

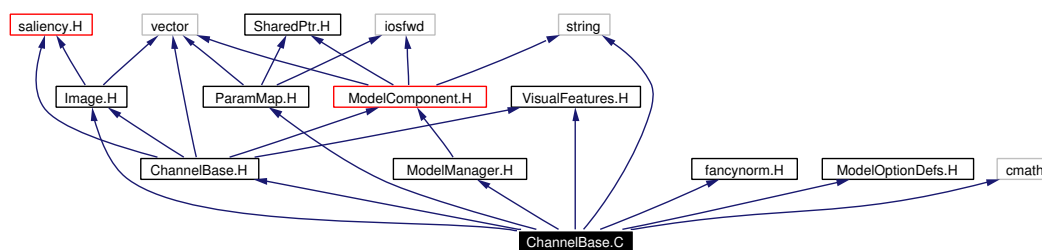
Container class for holding cascading energy values

Definition in file [cascadeHold.H](#).

## 11.65 ChannelBase.C File Reference

```
#include "ChannelBase.H"  
#include "Image.H"  
#include "ParamMap.H"  
#include "VisualFeatures.H"  
#include "fancynorm.H"  
#include "ModelOptionDefs.H"  
#include "ModelManager.H"  
#include <cmath>  
#include <string>
```

Include dependency graph for ChannelBase.C:



### 11.65.1 Detailed Description

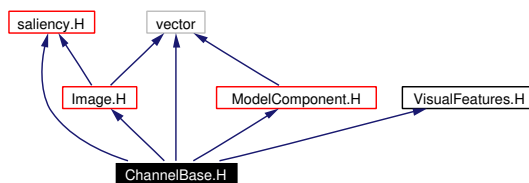
The base class for all channel classes

Definition in file [ChannelBase.C](#).

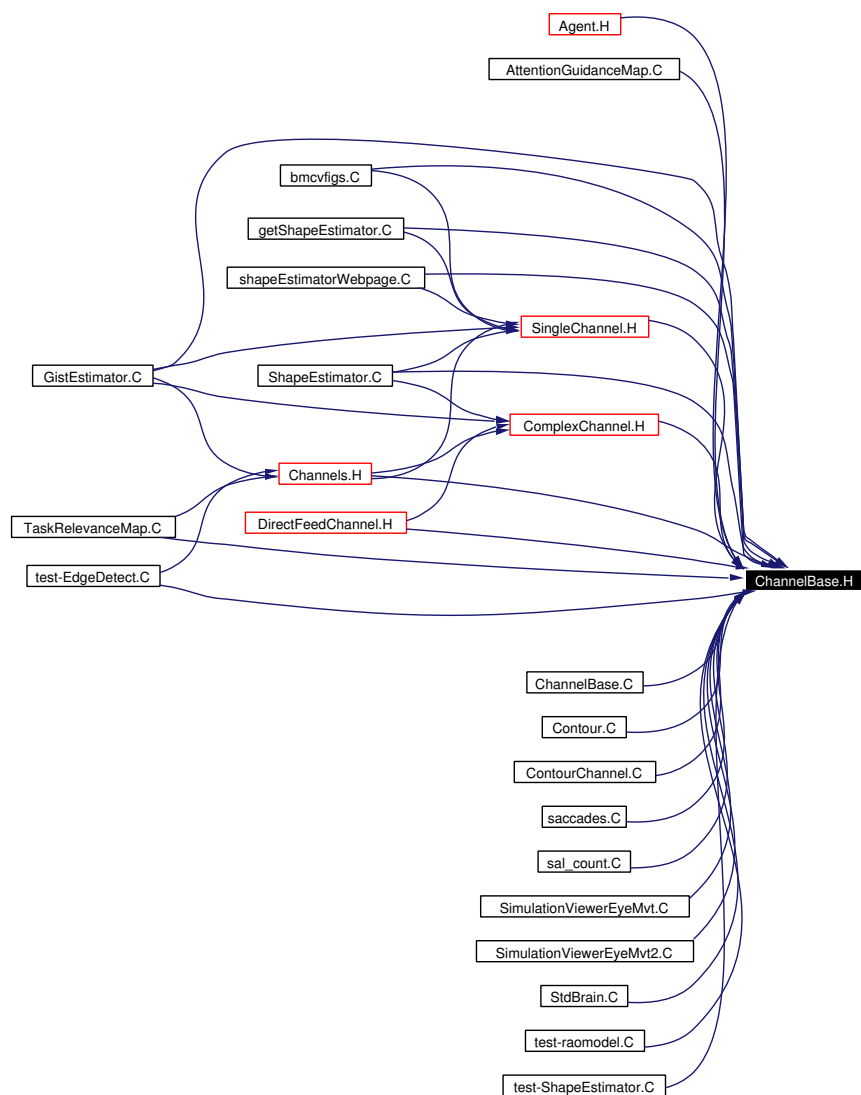
## 11.66 ChannelBase.H File Reference

```
#include "saliency.H"  
#include "VisualFeatures.H"  
#include "Image.H"  
#include <vector>  
#include "ModelComponent.H"
```

Include dependency graph for ChannelBase.H:



This graph shows which files directly or indirectly include this file:



## Classes

- class [ChannelListener](#)  
A callback class for use with [ChannelBase](#) and its derivatives.
- class [ChannelBase](#)  
*ChannelBase* represents the various computational modules in [VisualCortex](#).

### 11.66.1 Detailed Description

The base class for all channel classes

Definition in file [ChannelBase.H](#).

## 11.67 Channels.C File Reference

```
#include "Channels.H"
#include "ModelManager.H"
#include "Image_All.H"
#include "ImageSet.H"
#include "Jet.H"
#include "LevelSpec.H"
#include "log.H"
#include "saliency.H"
#include "ParamMap.H"
#include "PyrBuilder.H"
#include "Pyramid_Ops.H"
#include "ModelOptionDefs.H"
#include "StringConversions.H"
#include "Image_Transforms.H"
#include "Image_ColorOps.H"
#include "Image_FilterOps.H"
#include "colorDefs.H"
#include "ImageCache.H"
#include <cstdio>
#include <fstream>
#include <iomanip>
#include <iostream>
#include <limits>
#include <algorithm>
#include <cmath>
#include <numeric>
```

Include dependency graph for Channels.C:

## Defines

- `#define ACCUMCORR`

## Variables

- `const int pedX = 9`
- `const int pedY = 15`
- `const float pedFILT [pedX * pedY]`

### 11.67.1 Detailed Description

Feature channels for use in visual cortex classes

Definition in file [Channels.C](#).

### 11.67.2 Define Documentation

#### 11.67.2.1 `#define ACCUMCORR`

**Value:**

```
if (itsMap.coordsOk(pp) && itsMask.getVal(pp) == 0) \
{ ++n; itsMask.setVal(pp, 255); corr += 1.0 - \
  corrpitch(*bwimg, topleft1, patchdims, *bwimg, \
  Point2D(pp.i << lev, pp.j << lev)); }
```

Definition at line 2772 of file Channels.C.

### 11.67.3 Variable Documentation

#### 11.67.3.1 `const float pedFILT[pedX*pedY] [static]`

**Initial value:**

```
{
  -3.0, -3.0, -3.0, -3.0, -3.0, -3.0, -3.0, -3.0, -3.0,
  -3.0, -3.0, -3.0, 0.0, 1.0, 0.0, -3.0, -3.0, -3.0,
  -3.0, -3.0, 0.0, 2.0, 3.0, 2.0, 0.0, -3.0, -3.0,
  -3.0, 0.0, 0.0, 1.0, 3.0, 1.0, 0.0, 0.0, -3.0,
  -3.0, 1.0, 2.0, 3.0, 3.0, 3.0, 2.0, 1.0, -3.0,
  -3.0, 1.0, 2.0, 3.0, 3.0, 3.0, 2.0, 1.0, -3.0,
  -3.0, 1.0, 2.0, 3.0, 3.0, 3.0, 2.0, 1.0, -3.0,
  -3.0, 1.0, 2.0, 3.0, 3.0, 3.0, 2.0, 1.0, -3.0,
  -3.0, 1.0, 2.0, 3.0, 3.0, 3.0, 2.0, 1.0, -3.0,
```



```
-3.0,  0.0,  2.0,  3.0,  2.0,  3.0,  2.0,  0.0, -3.0,  
-3.0,  0.0,  2.0,  3.0,  0.0,  3.0,  2.0,  0.0, -3.0,  
-3.0,  0.0,  2.0,  3.0,  0.0,  3.0,  2.0,  0.0, -3.0,  
-3.0,  0.0,  2.0,  3.0,  0.0,  3.0,  2.0,  0.0, -3.0,  
-3.0,  0.0,  2.0,  3.0,  0.0,  3.0,  2.0,  0.0, -3.0,  
-3.0, -3.0, -3.0, -3.0, -3.0, -3.0, -3.0, -3.0, -3.0  
}
```

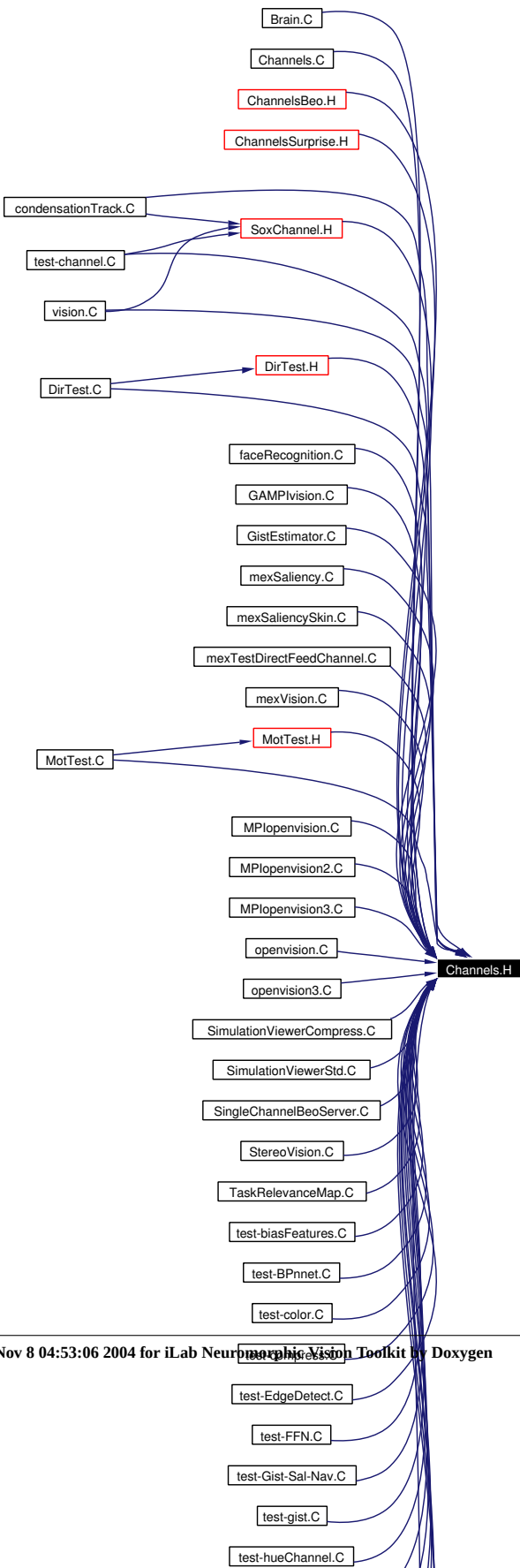
Definition at line 1634 of file Channels.C.

## 11.68 Channels.H File Reference

```
#include "ChannelBase.H"
#include "ComplexChannel.H"
#include "SingleChannel.H"
#include "Image.H"
#include "PyrBuilder.H"
#include "ImageSet_Ops.H"
#include "OrientComputeTypes.H"
#include "VisualCortex.H"
#include "VisualObject.H"
#include "ColorComputeTypes.H"
#include "ImageCache.H"
```

Include dependency graph for Channels.H:

This graph shows which files directly or indirectly include this file:



## Classes

- class [IntensityChannel](#)  
*An intensity channel.*
- class [ConvolveChannel](#)  
*A generic convolution channel.*
- class [RGBConvolveChannel](#)  
*A generic convolution channel using color filters.*
- class [RedGreenChannel](#)  
*A red-green double-opponent color channel.*
- class [BlueYellowChannel](#)  
*A blue-yellow double-opponent color channel.*
- class [SOREdGreenChannel](#)  
*Single Opponent [RedGreenChannel](#) member definitions:.*
- class [SOGreenRedChannel](#)  
*Single Opponent [GreenRedChannel](#) member definitions:.*
- class [SOBlueYellowChannel](#)  
*Single Opponent [BlueYellowChannel](#) member definitions:.*
- class [SOYellowBlueChannel](#)  
*Single Opponent [YellowBlueChannel](#) member definitions:.*
- class [RedChannel](#)  
*A red color channel.*
- class [GreenChannel](#)  
*A green color channel.*
- class [BlueChannel](#)  
*A blue color channel.*
- class [YellowChannel](#)  
*A yellow double-opponent color channel.*
- class [GaborChannel](#)  
*An orientation-tuned gabor channel.*

- class [FlickerChannel](#)  
*A temporal flicker channel.*
- class [ColorBandChannel](#)  
*A color band channel.*
- class [DirectionChannel](#)  
*a motion sensitive channel with direction selectivity*
- class [TemplateMatchChannel](#)  
*Convolves each level of its pyramid with a feature template.*
- class [PedestrianChannel](#)  
*A template-matching channel for detecting pedestrians.*
- class [JunctionChannel](#)  
*A junction channel.*
- class [EntropyChannel](#)  
*An entropy channel.*
- class [VarianceChannel](#)  
*A variance channel, computes the variance in an image patch.*
- class [TcorrChannel](#)  
*A temporal correlation channel.*
- class [ScorrChannel](#)  
*A spatial correlation channel.*
- class [InformationChannel](#)  
*An information channel.*
- class [DummyChannel](#)  
*A dummy channel that simply passes its greyscale input to its output.*
- class [PN03contrastChannel](#)  
*A Contrast channel (based on Parkhurst & Niebur 2003).*
- class [DisparityChannel](#)  
*An disparity channel.*

- class [HueChannel](#)  
*A hue similarity channel.*
- class [SkinHueChannel](#)  
*A channel for detecting skin color.*
- class [MultiConvolveChannel](#)  
*A generic multi-convolution channel.*
- class [CompositeColorChannel](#)  
*A composite color channel class that contains single and double !*
- class [ColorChannel](#)  
*A double opponent color channel class that combines r/g, b/y subchannels.*
- class [SOColorChannel](#)
- class [OrientationChannel](#)  
*A composite channel containing a set of orientation-tuned gabor channels.*
- class [MultiColorBandChannel](#)  
*A composite channel containing a set of color-band channels.*
- class [MotionChannel](#)  
*A composite channel containing a set of direction channels.*
- class [StereoChannel](#)  
*A composite channel containing a set of Disparity channels.*
- class [LJunctionChannel](#)  
*An L junction channel that detects L junctions at different.*
- class [TJunctionChannel](#)  
*A T junction channel that detects T junctions at different.*
- class [XJunctionChannel](#)  
*A X junction channel that detects crosses at different.*
- class [EndPointChannel](#)  
*An end point detector channel that detects line ends at different.*

### 11.68.1 Detailed Description

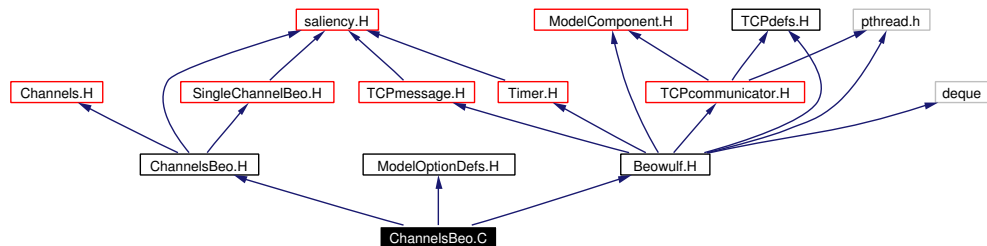
Feature channels for use in visual cortex classes

Definition in file [Channels.H](#).

## 11.69 ChannelsBeo.C File Reference

```
#include "ChannelsBeo.H"  
#include "ModelOptionDefs.H"  
#include "Beowulf.H"
```

Include dependency graph for ChannelsBeo.C:



### 11.69.1 Detailed Description

Definitions for the Beo version of the channels

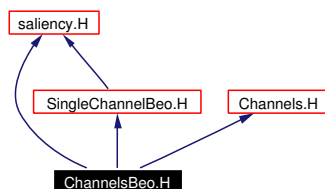
Definition in file [ChannelsBeo.C](#).



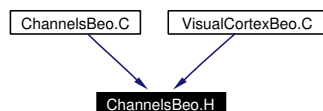
## 11.70 ChannelsBeo.H File Reference

```
#include "saliency.H"  
#include "SingleChannelBeo.H"  
#include "Channels.H"
```

Include dependency graph for ChannelsBeo.H:



This graph shows which files directly or indirectly include this file:



### Classes

- class [IntensityChannelBeo](#)  
*Beowulf version of [IntensityChannel](#).*
- class [RedGreenChannelBeo](#)  
*Beowulf version of [RedGreenChannel](#).*
- class [BlueYellowChannelBeo](#)  
*Beowulf version of [BlueYellowChannel](#).*
- class [GaborChannelBeo](#)  
*Beowulf version of [GaborChannel](#).*
- class [FlickerChannelBeo](#)  
*Beowulf version of [FlickerChannel](#).*
- class [ColorBandChannelBeo](#)

*Beowulf* version of [ColorBandChannel](#).

- class [DirectionChannelBeo](#)  
*Beowulf* version of [DirectionChannel](#).
- class [ColorChannelBeo](#)  
*Beowulf* version of [ColorChannel](#).
- class [OrientationChannelBeo](#)  
*Beowulf* version of [OrientationChannel](#).
- class [MultiColorBandChannelBeo](#)  
*Beowulf* version of [MultiColorBandChannel](#).
- class [MotionChannelBeo](#)  
*Beowulf* version of [MotionChannel](#).

### 11.70.1 Detailed Description

Definitions for the Beo version of the channels

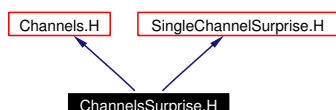
Definition in file [ChannelsBeo.H](#).

## 11.71 ChannelsSurprise.H File Reference

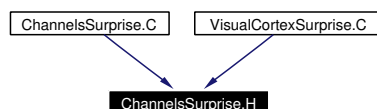
```
#include "Channels.H"
```

```
#include "SingleChannelSurprise.H"
```

Include dependency graph for ChannelsSurprise.H:



This graph shows which files directly or indirectly include this file:



### Classes

- class [IntensityChannelSurprise](#)  
*An intensity channel.*
- class [ConvolveChannelSurprise](#)  
*A generic convolution channel.*
- class [RedGreenChannelSurprise](#)  
*A red-green double-opponent color channel.*
- class [BlueYellowChannelSurprise](#)  
*A blue-yellow double-opponent color channel.*
- class [SORedGreenChannelSurprise](#)  
*Single Opponent [RedGreenChannelSurprise](#) member definitions:.*
- class [SOGreenRedChannelSurprise](#)  
*Single Opponent [GreenRedChannelSurprise](#) member definitions:.*
- class [SOBlueYellowChannelSurprise](#)

Single Opponent [BlueYellowChannelSurprise](#) member definitions:.

- class [SOYellowBlueChannelSurprise](#)  
Single Opponent YellowBlueChannelSurprise member definitions:.
- class [RedChannelSurprise](#)  
A red color channel.
- class [GreenChannelSurprise](#)  
A green color channel.
- class [BlueChannelSurprise](#)  
A blue color channel.
- class [YellowChannelSurprise](#)  
A yellow double-opponent color channel.
- class [GaborChannelSurprise](#)  
An orientation-tuned gabor channel.
- class [FlickerChannelSurprise](#)  
A temporal flicker channel.
- class [ColorBandChannelSurprise](#)  
A color band channel.
- class [DirectionChannelSurprise](#)  
a motion sensitive channel with direction selectivity
- class [TemplateMatchChannelSurprise](#)  
Convolves each level of its pyramid with a feature template.
- class [PedestrianChannelSurprise](#)  
A template-matching channel for detecting pedestrians.
- class [JunctionChannelSurprise](#)  
A junction channel.
- class [DummyChannelSurprise](#)  
A dummy channel.
- class [MultiConvolveChannelSurprise](#)

*A generic multi-convolution channel.*

- class [CompositeColorChannelSurprise](#)  
*A composite color channel class that contains single and double opp.*
- class [ColorChannelSurprise](#)  
*A double opponent color channel class that combines r/g, b/y subchannels.*
- class [SOColorChannelSurprise](#)  
*A single opponent color channel class that combines r/g, g/r, b/y, etc.*
- class [OrientationChannelSurprise](#)  
*A composite channel containing a set of orientation-tuned gabor channels.*
- class [MultiColorBandChannelSurprise](#)  
*A composite channel containing a set of color-band channels.*
- class [MotionChannelSurprise](#)  
*A composite channel containing a set of direction channels.*
- class [LJunctionChannelSurprise](#)  
*An L junction channel that detects L junctions at different orientations.*
- class [TJunctionChannelSurprise](#)  
*A T junction channel that detects T junctions at different orientations.*
- class [XJunctionChannelSurprise](#)  
*A X junction channel that detects crosses at different orientations.*
- class [EndPointChannelSurprise](#)  
*An end point detector channel that detects line ends at different.*

### 11.71.1 Detailed Description

Feature channels, surprise version

Definition in file [ChannelsSurprise.H](#).

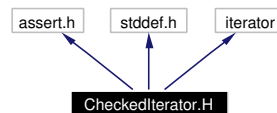
## 11.72 CheckedIterator.H File Reference

```
#include <assert.h>
```

```
#include <stddef.h>
```

```
#include <iterator>
```

Include dependency graph for CheckedIterator.H:



### Classes

- class [CheckedIterator](#)  
*A range-checked iterator class for memory debugging.*

### Typedefs

- typedef unsigned long **diff\_t**

#### 11.72.1 Detailed Description

A range-checked iterator

Definition in file [CheckedIterator.H](#).

## 11.73 CINNIC.C File Reference

```
#include "CINNIC.H"  
#include "Image_ColorOps.H"  
#include "Image_FilterOps.H"  
#include "Image_ShapeOps.H"  
#include "Timer.H"  
#include <fstream>  
#include <cmath>
```

Include dependency graph for CINNIC.C:



### 11.73.1 Detailed Description

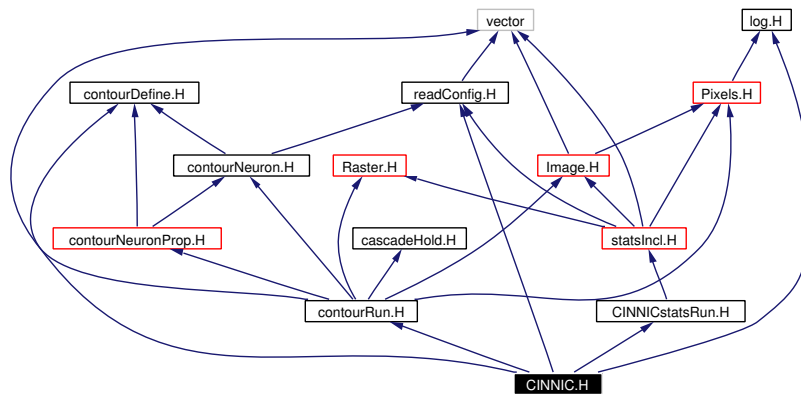
[CINNIC](#) classes

Definition in file [CINNIC.C](#).

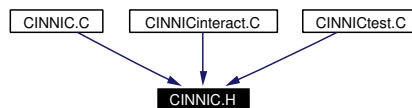
## 11.74 CINNIC.H File Reference

```
#include "contourRun.H"
#include "readConfig.H"
#include "log.H"
#include "contourDefine.H"
#include "CINNICstatsRun.H"
```

Include dependency graph for CINNIC.H:



This graph shows which files directly or indirectly include this file:



### Classes

- class [CINNIC](#)  
*Main class to run an image in CINNICtest for contours.*

### 11.74.1 Detailed Description

Test binary for [CINNIC](#) neuron

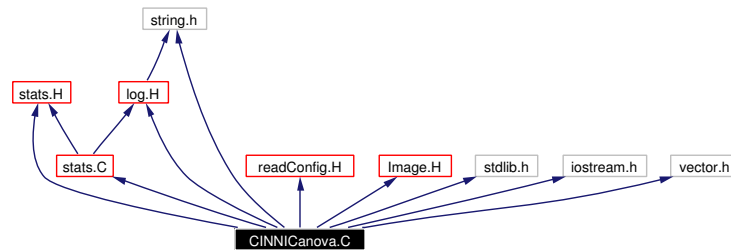


Definition in file [CINNIC.H](#).

## 11.75 CINNICanova.C File Reference

```
#include "log.H"
#include "readConfig.H"
#include "Image.H"
#include "stats.H"
#include "stats.C"
#include <stdlib.h>
#include <iostream.h>
#include <string.h>
#include <vector.h>
```

Include dependency graph for CINNICanova.C:



### Functions

- `int main (int argc, char *argv[ ])`

### Variables

- `char * savefilename`  
*the file to save stats out to*
- `readConfig ReadConfigA (25)`  
*two data sets to open*
- `readConfig ReadConfigB (25)`  
*two data sets to open*

- char \* [dataSet1](#)  
*Files containing data sets.*
- char \* **dataSet2**
- vector< float > [rSet1](#)  
*Vector for regresion data set 1.*
- vector< float > [rSet2](#)  
*Vector for regresion data set 2.*
- vector< float > [eSet1](#)  
*Vector for euclidian data set 1.*
- vector< float > [eSet2](#)  
*Vector for euclidian data set 2.*
- stats< float > [rStats](#)  
*Stats objects.*
- stats< float > [eStats](#)  
*Stats objects.*
- float [rF](#)  
*F from anova.*
- float [eF](#)  
*F from anova.*

### 11.75.1 Detailed Description

do ANOVA tests on [CINNIC](#) outputs

Definition in file [CINNICanova.C](#).

### 11.75.2 Variable Documentation

#### 11.75.2.1 char\* [dataSet1](#)

Files containing data sets.

Definition at line 29 of file CINNICanova.C.

**11.75.2.2 float eF**

F from anova.

Definition at line 44 of file CINNICanova.C.

**11.75.2.3 vector<float> eSet1**

Vector for euclidian data set 1.

Definition at line 36 of file CINNICanova.C.

**11.75.2.4 vector<float> eSet2**

Vector for euclidian data set 2.

Definition at line 38 of file CINNICanova.C.

**11.75.2.5 stats<float> eStats**

Stats objects.

Definition at line 42 of file CINNICanova.C.

**11.75.2.6 readConfig ReadConfigA(25)**

two data sets to open

**11.75.2.7 readConfig ReadConfigB(25)**

two data sets to open

**11.75.2.8 float rF**

F from anova.

Definition at line 44 of file CINNICanova.C.

**11.75.2.9 vector<float> rSet1**

Vector for regresion data set 1.

Definition at line 32 of file CINNICanova.C.

**11.75.2.10** `vector<float> rSet2`

Vector for regresion data set 2.

Definition at line 34 of file CINNICanova.C.

**11.75.2.11** `stats<float> rStats`

Stats objects.

Definition at line 40 of file CINNICanova.C.

**11.75.2.12** `char* savefilename`

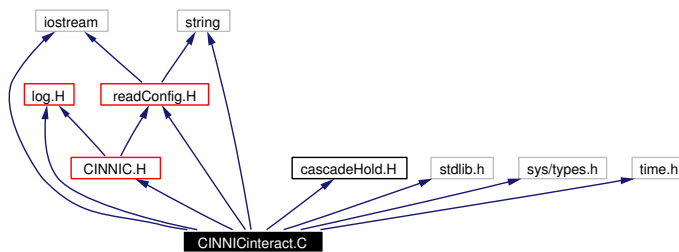
the file to save stats out to

Definition at line 23 of file CINNICanova.C.

## 11.76 CINNICInteract.C File Reference

```
#include "log.H"
#include "cascadeHold.H"
#include "readConfig.H"
#include "CINNIC.H"
#include <stdlib.h>
#include <iostream>
#include <string>
#include <sys/types.h>
#include <time.h>
```

Include dependency graph for CINNICInteract.C:



### Functions

- int [main](#) (int argc, char \*argv[ ])  
*This is the main function for [CINNIC](#).*

### Variables

- [ContourNeuronCreate](#) < float > [CINNICreate](#)  
*Create a [CINNIC](#) neuron, use float.*
- [CINNIC skeptic](#)  
*Standard [CINNIC](#) test object.*
- istream \* [argInput](#)

*pointer to command line argument stream*

- `char * configFile`

*This is the configFile name.*

- `readConfig configIn (25)`

*This is the configFile object.*

## 11.76.1 Detailed Description

Interactive [CINNIC](#)

Definition in file [CINNICinteract.C](#).

## 11.76.2 Function Documentation

### 11.76.2.1 `int main (int argc, char * argv[ ])`

This is the main function for [CINNIC](#).

to use this you must have a config file called contour.conf you should run this file by running it from src2 and executing it as such "../bin/CINNICTest ../image\_in/imagefile outputname" where imagefile is a ppm file (do not include extention) and outputname is the name you would like attached to all the output files.

Definition at line 44 of file CINNICinteract.C.

References [argInput](#), [CINNICreate](#), [configFile](#), [configIn](#), [CINNIC::convolve-Test\(\)](#), [ContourNeuronCreate< float >::CreateNeuron\(\)](#), [CINNIC::filename](#), [main\(\)](#), [CINNIC::Ninput](#), [NO\\_INIT](#), [readConfig::openFile\(\)](#), [readConfig::readFileTrue\(\)](#), [readConfig::readFileValueF\(\)](#), [readConfig::readFileValueName\(\)](#), [Raster::ReadGray\(\)](#), [CINNIC::savefilename](#), and [skeptc](#).

## 11.76.3 Variable Documentation

### 11.76.3.1 `istream* argInput`

*pointer to command line argument stream*

Definition at line 31 of file CINNICinteract.C.

Referenced by [main\(\)](#).

**11.76.3.2   [ContourNeuronCreate](#)<float> [CINNICreate](#)**

Create a [CINNIC](#) neuron, use float.

Definition at line 27 of file CINNICinteract.C.

Referenced by main().

**11.76.3.3   char\* [configFile](#)**

This is the configFile name.

Definition at line 33 of file CINNICinteract.C.

Referenced by main().

**11.76.3.4   [readConfig](#) [configIn](#)(25)**

This is the configFile object.

Referenced by main().

**11.76.3.5   [CINNIC](#) [skeptic](#)**

Standard [CINNIC](#) test object.

Definition at line 29 of file CINNICinteract.C.

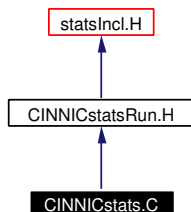
Referenced by main().



## 11.77 CINNICstats.C File Reference

```
#include "CINNICstatsRun.H"
```

Include dependency graph for CINNICstats.C:



### Functions

- int [main](#) (int argc, char \*argv[ ])  
*This is a package to run tests on the output from [CINNIC](#).*

### Variables

- char \* [savefilename](#)  
*the file to save stats out to*
- char \* [listfile](#)  
*this string contains the name of the list file to open of images*
- char \* [configFile](#)  
*this contains the name of the config file*
- char \* [configFile2](#)  
*this contains the name of the config file*
- char \* [specialParam](#)  
*this is the parameter special for this analysis*
- [readConfig fileList](#) (25)  
*This is the configFile object.*
- [readConfig ReadConfigMain](#) (25)

*this is the main config file*

- [readConfig](#) [ReadConfigMain2](#) (25)

*this is the main config file*

- CINNICstatsRun [runStats](#)

*main object for stats*

### 11.77.1 Detailed Description

do statistical tests on [CINNIC](#) outputs

Definition in file [CINNICstats.C](#).

### 11.77.2 Function Documentation

#### 11.77.2.1 `int main (int argc, char * argv[ ])`

This is a package to run tests on the output from [CINNIC](#).

main object for stats

Definition at line 69 of file CINNICstats.C.

References `ASSERT`, `configFile`, `configFile2`, `fileList`, `readConfig::getItemValueF()`, `listfile`, `main()`, `readConfig::openFile()`, `ReadConfigMain`, `ReadConfigMain2`, `runStats`, and `specialParam`.

### 11.77.3 Variable Documentation

#### 11.77.3.1 `char* configFile`

this contains the name of the config file

Definition at line 54 of file CINNICstats.C.

#### 11.77.3.2 `char* configFile2`

this contains the name of the config file

Definition at line 56 of file CINNICstats.C.

Referenced by `main()`.

### 11.77.3.3 `readConfig fileList(25)`

This is the configFile object.

Referenced by main().

### 11.77.3.4 `char* listfile`

this string contains the name of the list file to open of images

Definition at line 52 of file CINNICstats.C.

Referenced by main().

### 11.77.3.5 `readConfig ReadConfigMain(25)`

this is the main config file

Referenced by main().

### 11.77.3.6 `readConfig ReadConfigMain2(25)`

this is the main config file

Referenced by main().

### 11.77.3.7 `CINNICstatsRun runStats`

main object for stats

Definition at line 66 of file CINNICstats.C.

Referenced by main().

### 11.77.3.8 `char* savefilename`

the file to save stats out to

Definition at line 50 of file CINNICstats.C.

### 11.77.3.9 `char* specialParam`

this is the parameter special for this analysis

Definition at line 58 of file CINNICstats.C.

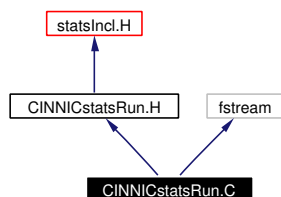
Referenced by main().

## 11.78 CINNICstatsRun.C File Reference

```
#include "CINNICstatsRun.H"
```

```
#include <fstream>
```

Include dependency graph for CINNICstatsRun.C:



### 11.78.1 Detailed Description

run anal. of [CINNIC](#)

Definition in file [CINNICstatsRun.C](#).

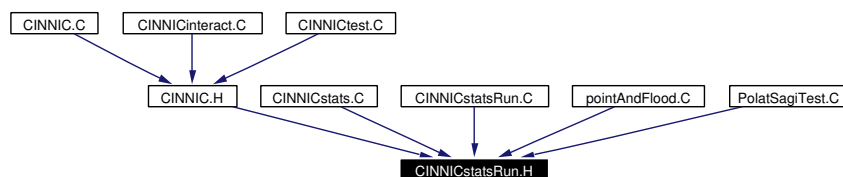
## 11.79 CINNICstatsRun.H File Reference

```
#include "statsIncl.H"
```

Include dependency graph for CINNICstatsRun.H:



This graph shows which files directly or indirectly include this file:



### Classes

- class **CINNICstatsRun**

#### 11.79.1 Detailed Description

Run some stats with [CINNIC](#)

Definition in file [CINNICstatsRun.H](#).

## 11.80 CINNICtest.C File Reference

```
#include "log.H"
#include "cascadeHold.H"
#include "readConfig.H"
#include "CINNIC.H"
#include <stdlib.h>
#include <iostream>
#include <string>
#include <sys/types.h>
#include <time.h>
```

Include dependency graph for CINNICtest.C:

### Functions

- int [main](#) (int argc, char \*argv[ ])  
*This is the main function for [CINNIC](#).*

### Variables

- [ContourNeuronCreate](#)< float > [CINNICreate](#)  
*Create a [CINNIC](#) neuron, use float.*
- [CINNIC](#) skeptic  
*Standard [CINNIC](#) test object.*
- istream \* [argInput](#)  
*pointer to command line argument stream*
- char \* [configFile](#)  
*This is the configFile name.*
- [readConfig](#) [configIn](#) (25)  
*This is the configFile object.*

### 11.80.1 Detailed Description

Tests the [CINNIC](#) neuron

Definition in file [CINNICtest.C](#).

### 11.80.2 Function Documentation

#### 11.80.2.1 `int main (int argc, char * argv[ ])`

This is the main function for [CINNIC](#).

to use this you must have a config file called contour.conf you should run this file by running it from src3 and executing it as such `"../bin/CINNICtest ../image_in/imagefile outputname"` where imagefile is a ppm file (do not include extension) and outputname is the name you would like attached to all the output files.

Definition at line 75 of file CINNICtest.C.

References [PropHold::ang](#), [PropHold::ang2](#), [PropHold::angAB](#), [PropHold::angABD](#), [argInput](#), [CINNICCreate](#), [PropHold::coLinear](#), [configFile](#), [configIn](#), [ContourNeuronCreate< float >::CreateNeuron\(\)](#), [PropHold::dis](#), [CINNIC::filename](#), [ContourNeuronCreate< float >::FourDNeuralMap](#), [main\(\)](#), [NeuralAngles](#), [CINNIC::Ninput](#), [NO\\_INIT](#), [readConfig::openFile\(\)](#), [PropHold::pol](#), [Raster::ReadGray\(\)](#), [CINNIC::RunSimpleImage\(\)](#), [CINNIC::savefilename](#), [PropHold::sender](#), [skeptical](#), and [CINNIC::viewNeuronTemplate\(\)](#).

### 11.80.3 Variable Documentation

#### 11.80.3.1 `istream* argInput`

pointer to command line argument stream

Definition at line 62 of file CINNICtest.C.

#### 11.80.3.2 `ContourNeuronCreate<float> CINNICCreate`

Create a [CINNIC](#) neuron, use float.

Definition at line 58 of file CINNICtest.C.

#### 11.80.3.3 `char* configFile`

This is the configFile name.

Definition at line 64 of file CINNICtest.C.

**11.80.3.4** [readConfig configIn\(25\)](#)

This is the configFile object.

**11.80.3.5** [CINNIC skeptic](#)

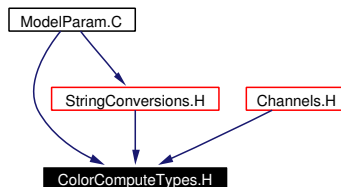
Standard [CINNIC](#) test object.

Definition at line 60 of file CINNICtest.C.



## 11.81 ColorComputeTypes.H File Reference

This graph shows which files directly or indirectly include this file:



### Defines

- `#define NBCOLORCOMPUTETYPES 2`  
*number of known OrientComputeTypes:*

### Enumerations

- `enum ColorComputeType { COLstandard = 0, COLsimple = 1 }`  
*Possible types computing orientations.*

### Functions

- `const char * colorComputeTypeName (const ColorComputeType p)`

#### 11.81.1 Detailed Description

Definition in file [ColorComputeTypes.H](#).

#### 11.81.2 Define Documentation

##### 11.81.2.1 `#define NBCOLORCOMPUTETYPES 2`

number of known OrientComputeTypes:

Definition at line 49 of file ColorComputeTypes.H.

Referenced by `convertFromString()`.

### 11.81.3 Enumeration Type Documentation

#### 11.81.3.1 enum [ColorComputeType](#)

Possible types computing orientations.

**Enumeration values:**

***COLstandard*** use the standard way of computing RG and BY

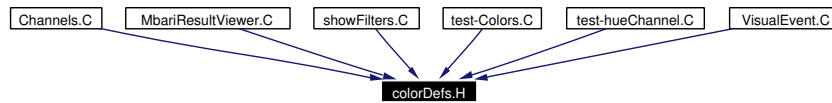
***COLsimple*** use the straight-forward simple way to do it

Definition at line 42 of file ColorComputeTypes.H.

Referenced by ColorChannel::ColorChannel(), and convertFromString().

## 11.82 colorDefs.H File Reference

This graph shows which files directly or indirectly include this file:



### Defines

- `#define COL_BLACK PixRGB<byte>( 0, 0, 0)`
- `#define COL_WHITE PixRGB<byte>(255,255,255)`
- `#define COL_YELLOW PixRGB<byte>(255,255, 0)`
- `#define COL_ORANGE PixRGB<byte>(255,128, 0)`
- `#define COL_BLUE PixRGB<byte>( 0, 0,255)`
- `#define COL_RED PixRGB<byte>(255, 0, 0)`
- `#define COL_GREEN PixRGB<byte>( 0,255, 0)`
- `#define COL_SKIN PixRGB<byte>(176,122,107)`
- `#define COL_SKIN_MUR 0.434904`
- `#define COL_SKIN_MUG 0.301983`
- `#define COL_SKIN_SIGR 0.053375`
- `#define COL_SKIN_SIGG 0.024349`
- `#define COL_SKIN_RHO 0.5852`
- `#define COL_TRANSPARENT PixRGB<byte>( 1, 1, 1)`
- `#define COL_INTERESTING COL_YELLOW`
- `#define COL_CANDIDATE COL_ORANGE`
- `#define COL_SKIP COL_BLUE`
- `#define COL_PREDICTION COL_GREEN`
- `#define COL_FOE COL_RED`

### 11.82.1 Detailed Description

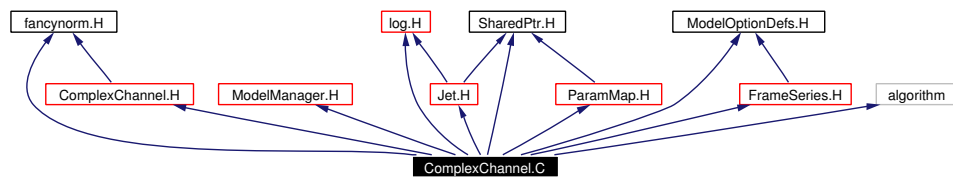
definitions of a few colors

Definition in file [colorDefs.H](#).

## 11.83 ComplexChannel.C File Reference

```
#include "ComplexChannel.H"  
#include "ModelManager.H"  
#include "fancynorm.H"  
#include "log.H"  
#include "ParamMap.H"  
#include "SharedPtr.H"  
#include "Jet.H"  
#include "FrameSeries.H"  
#include "ModelOptionDefs.H"  
#include <algorithm>
```

Include dependency graph for ComplexChannel.C:



### 11.83.1 Detailed Description

Channel class that pools across multiple subchannels.

Definition in file [ComplexChannel.C](#).

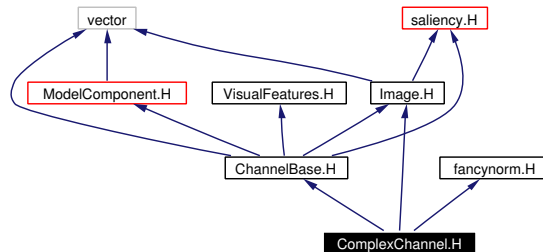
## 11.84 ComplexChannel.H File Reference

```
#include "ChannelBase.H"
```

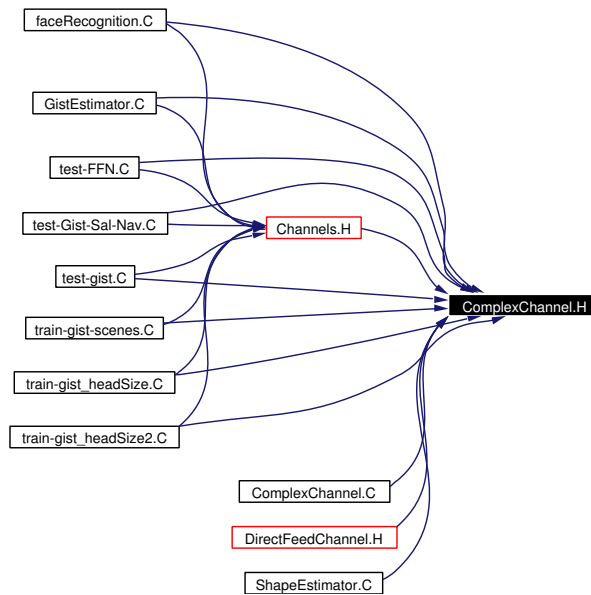
```
#include "Image.H"
```

```
#include "fancynorm.H"
```

Include dependency graph for ComplexChannel.H:



This graph shows which files directly or indirectly include this file:



### Classes

- class [ComplexChannel](#)

*A an abstract channel class that pools across several subchannels.*

### **11.84.1 Detailed Description**

Channel class that pools across multiple subchannels.

Definition in file [ComplexChannel.H](#).

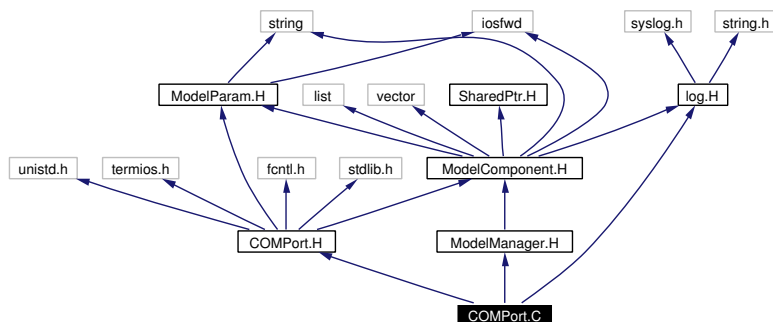
## 11.85 COMPort.C File Reference

```
#include "COMPort.H"
```

```
#include "log.H"
```

```
#include "ModelManager.H"
```

Include dependency graph for COMPort.C:



### 11.85.1 Detailed Description

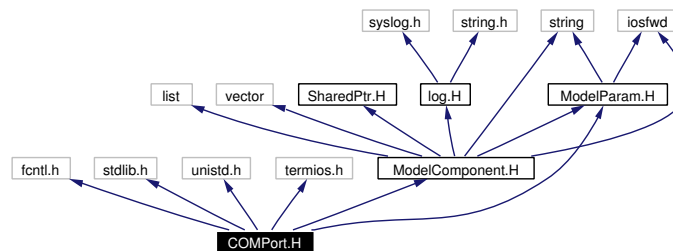
a simple class for interfacing with a serial port

Definition in file [COMPort.C](#).

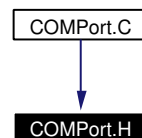
## 11.86 COMPort.H File Reference

```
#include <fcntl.h>
#include <stdlib.h>
#include <unistd.h>
#include <termios.h>
#include "ModelParam.H"
#include "ModelComponent.H"
```

Include dependency graph for COMPort.H:



This graph shows which files directly or indirectly include this file:



### Classes

- class [COMPort](#)  
*A simple class for interfacing with a serial (COM) port.*

### Defines

- #define **STOP\_BITS1** 0
- #define **STOP\_BITS2** CSTOPB



- #define **NO\_PARITY** 0
- #define **EVEN\_PARITY** PARENB
- #define **ODD\_PARITY** (PARENB|PARODD)

### 11.86.1 Detailed Description

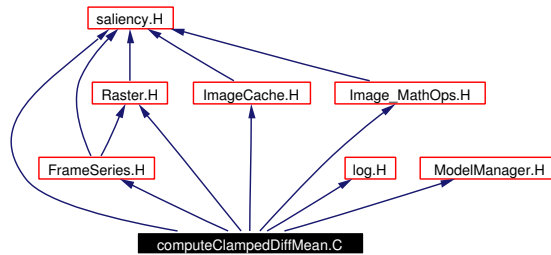
a simple class for interfacing with a serial port

Definition in file [COMPort.H](#).

## 11.87 computeClampedDiffMean.C File Reference

```
#include "saliency.H"
#include "log.H"
#include "FrameSeries.H"
#include "ModelManager.H"
#include "Raster.H"
#include "ImageCache.H"
#include "Image_MathOps.H"
```

Include dependency graph for computeClampedDiffMean.C:



### Functions

- `int main (const int argc, const char **argv)`

#### 11.87.1 Detailed Description

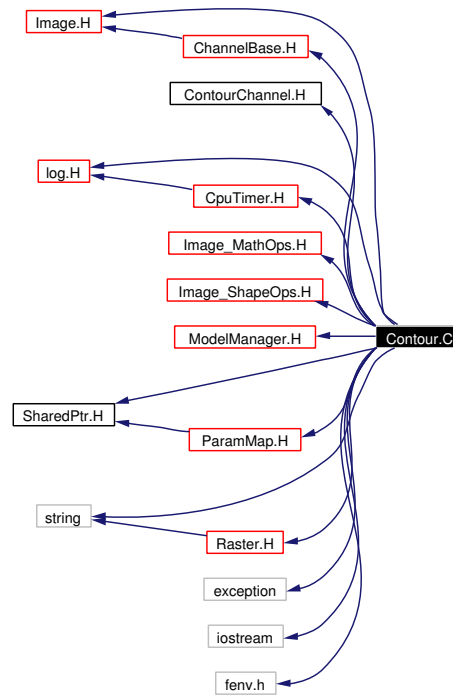
Executable that computes the `absDiffMean` of a number of images

Definition in file [computeClampedDiffMean.C](#).

## 11.88 Contour.C File Reference

```
#include "ChannelBase.H"  
#include "ContourChannel.H"  
#include "CpuTimer.H"  
#include "Image.H"  
#include "Image_MathOps.H"  
#include "Image_ShapeOps.H"  
#include "log.H"  
#include "ModelManager.H"  
#include "ParamMap.H"  
#include "Raster.H"  
#include "SharedPtr.H"  
#include <exception>  
#include <iostream>  
#include <string>  
#include <fenv.h>
```

Include dependency graph for Contour.C:



## Functions

- [Point2D](#) **topLeft** (const [Rectangle](#) &r)
- [Dims](#) **dimsOf** (const [Rectangle](#) &r)
- [Image](#)< float > **processImage** (const [Image](#)< byte > &img, [ModelManager](#) &manager, const char \*saveprefix, const char \*pmapfile)
- [Image](#)< float > **processWholeImage** (const [Image](#)< byte > &img, [ModelManager](#) &manager, const char \*saveprefix, const char \*pmapfile, int &pixused)
- [Image](#)< float > **processSplitImage** (const [Image](#)< byte > &img, [ModelManager](#) &manager, const char \*saveprefix, const char \*pmapfile, int &pixused)
- [Image](#)< float > **processImage** (const [Image](#)< byte > &img, [ModelManager](#) &manager, const char \*saveprefix, const char \*pmapfile, int &pixused)
- [Image](#)< float > **processWholeImage** (const [Image](#)< byte > &img, [ModelManager](#) &manager, const char \*saveprefix, const char \*pmapfile, int &pixused)
- [Image](#)< float > **processSplitImage** (const [Image](#)< byte > &input, [ModelManager](#) &manager, const char \*saveprefix, const char \*pmapfile, int &pixused)

- int **main** (int argc, char \*argv[ ])

## Variables

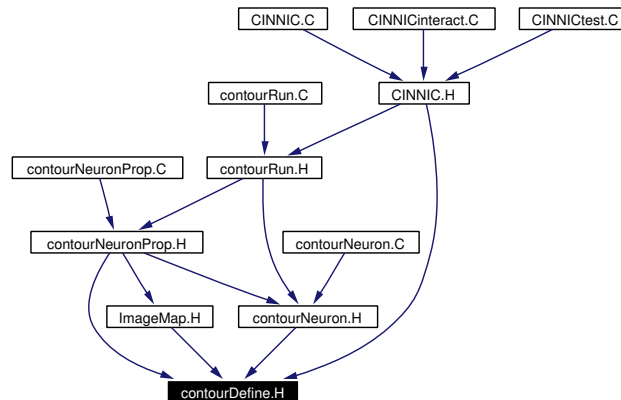
- int **TEMP\_COUNTER** = 0
- double **OVERLAP\_FRACTION** = 0.2
- int **SPLIT\_SIZE** = 1400000
- int **NEST\_DEPTH** = 0

### 11.88.1 Detailed Description

Definition in file [Contour.C](#).

## 11.89 contourDefine.H File Reference

This graph shows which files directly or indirectly include this file:



### Defines

- `#define ImageSizeX 100`
- `#define ImageSizeY 100`
- `#define XSize 12`
- `#define YSize 12`
- `#define AnglesUsed 12`
- `#define AngleCenter 90`
- `#define pi 3.14159`
- `#define dumbImageSize 100`
- `#define CutOffLevel .9`

### 11.89.1 Detailed Description

[CINNIC](#) classes

Definition in file [contourDefine.H](#).

## 11.90 contourNeuron.C File Reference

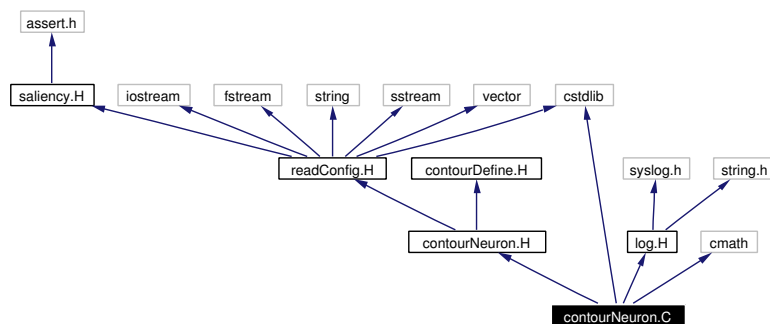
```
#include "contourNeuron.H"
```

```
#include "log.H"
```

```
#include <cmath>
```

```
#include <cstdlib>
```

Include dependency graph for contourNeuron.C:



### Variables

- float **AngleDropOff**
- float **AngleSupress**
- float **SupMult**
- float **OrthA**
- float **OrthB**
- float **OrthMult**
- float **valueCutOff**
- float **coLinearCutOff**
- float **NeuronExSize**
- float **NeuronSupSize**
- float **NeuronSupStart**
- float **NeuronOrthSize**
- float **CoLinearDiff**
- float **NeuralAngles** [AnglesUsed] = {0,15,30,45,60,75,90,105,120,135,150,165}

*How many angles are detected via gabor filter.*

- float **XCenter**
- float **YCenter**

- float **ExReach**
- float **SupReach**
- float **OrthReach**
- float **linearFactor2e**
- float **linearFactor3e**
- float **linearFactor2s**
- float **linearFactor3s**
- float **distanceSupFactor2**
- float **distanceSupFactor3**
- float **distanceExFactor2**
- float **distanceExFactor3**

### 11.90.1 Detailed Description

[CINNIC](#) classes

Definition in file [contourNeuron.C](#).

### 11.90.2 Variable Documentation

**11.90.2.1** float [NeuralAngles](#)[AnglesUsed] =  
{0,15,30,45,60,75,90,105,120,135,150,165}

How many angles are detected via gabor filter.

Definition at line 58 of file [contourNeuron.C](#).

Referenced by [ContourNeuronCreate](#)< TC2 >::CreateNeuron(), and [main\(\)](#).

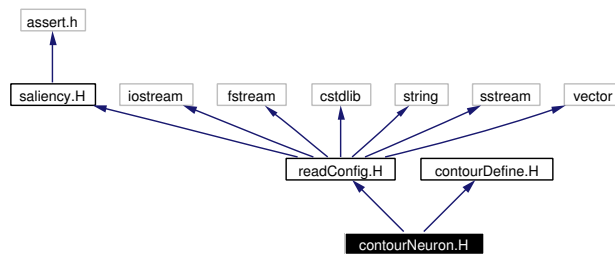


## 11.91 contourNeuron.H File Reference

```
#include "readConfig.H"
```

```
#include "contourDefine.H"
```

Include dependency graph for contourNeuron.H:



This graph shows which files directly or indirectly include this file:

### Classes

- class [PropHold](#)  
*this is a container class for the property at each hyper pixel for [ContourNeuronCreate](#)*
- class [ContourNeuronCreate](#)  
*This class will create the neuron connection template.*

### Variables

- float **AngleDropOff**
- float **AngleSupress**
- float **SupMult**
- float **OrthA**
- float **OrthB**
- float **OrthMult**
- float **valueCutOff**
- float **coLinearCutOff**
- float **NeuronExSize**
- float **NeuronSupSize**
- float **NeuronSupStart**
- float **NeuronOrthSize**

- float **CoLinearDiff**
- float [NeuralAngles](#) [AnglesUsed]

*How many angles are detected via gabor filter.*

- float **XCenter**
- float **YCenter**
- float **ExReach**
- float **SupReach**
- float **OrthReach**

### 11.91.1 Detailed Description

[CINNIC](#) classes

Definition in file [contourNeuron.H](#).

### 11.91.2 Variable Documentation

#### 11.91.2.1 float [NeuralAngles](#)[AnglesUsed]

How many angles are detected via gabor filter.

Definition at line 58 of file [contourNeuron.H](#).

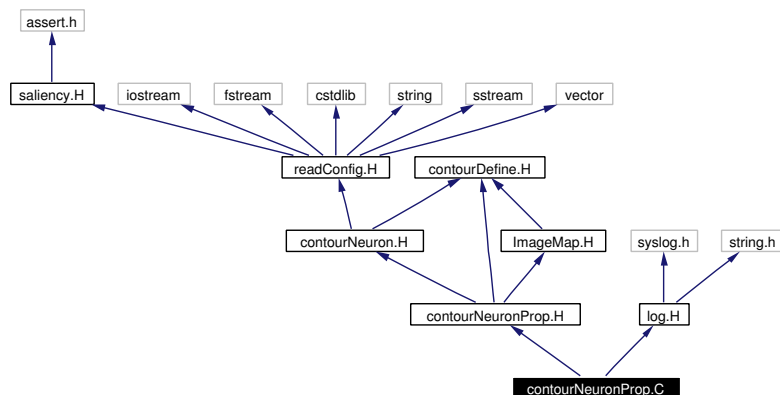
Referenced by `ContourNeuronCreate< TC2 >::CreateNeuron()`, and `main()`.

## 11.92 contourNeuronProp.C File Reference

```
#include "contourNeuronProp.H"
```

```
#include "log.H"
```

Include dependency graph for contourNeuronProp.C:



### 11.92.1 Detailed Description

[CINN](#) classes

Definition in file [contourNeuronProp.C](#).

## 11.93 contourNeuronProp.H File Reference

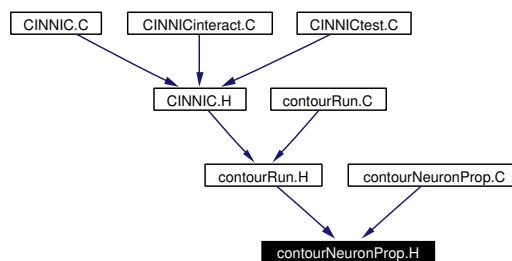
```
#include "contourNeuron.H"
```

```
#include "ImageMap.H"
```

```
#include "contourDefine.H"
```

Include dependency graph for contourNeuronProp.H:

This graph shows which files directly or indirectly include this file:



### Classes

- class [ContourNeuronProp](#)  
*Holds the property for each neuron.*

### 11.93.1 Detailed Description

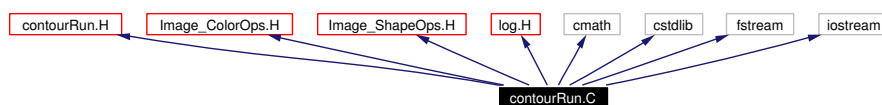
[CINNIC](#) classes

Definition in file [contourNeuronProp.H](#).

## 11.94 contourRun.C File Reference

```
#include "contourRun.H"  
#include "Image_ColorOps.H"  
#include "Image_ShapeOps.H"  
#include "log.H"  
#include <cmath>  
#include <cstdlib>  
#include <fstream>  
#include <iostream>
```

Include dependency graph for contourRun.C:



### Variables

- float **Resistance**

#### 11.94.1 Detailed Description

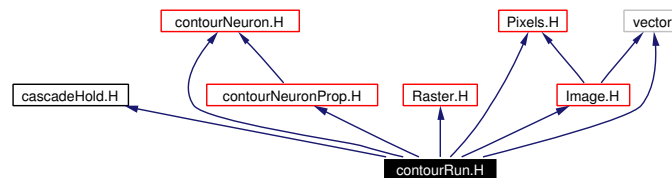
[CINNIC](#) classes - src3

Definition in file [contourRun.C](#).

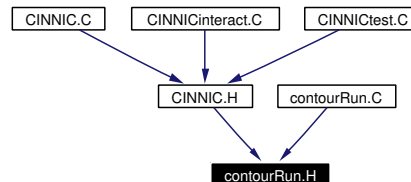
## 11.95 contourRun.H File Reference

```
#include "cascadeHold.H"
#include "contourNeuron.H"
#include "contourNeuronProp.H"
#include "Raster.H"
#include "Pixels.H"
#include "Image.H"
#include <vector>
```

Include dependency graph for contourRun.H:



This graph shows which files directly or indirectly include this file:



### Classes

- class [contourRun](#)

*Run the hyper column on a given image with a given connection template.*

### Defines

- #define **initSize** 50

### 11.95.1 Detailed Description

[CINNIC](#) classes

Definition in file [contourRun.H](#).

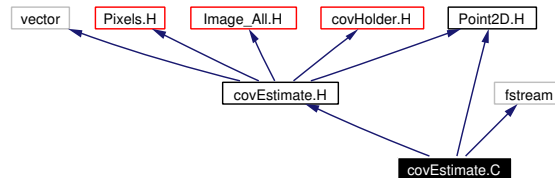
## 11.96 covEstimate.C File Reference

```
#include "covEstimate.H"
```

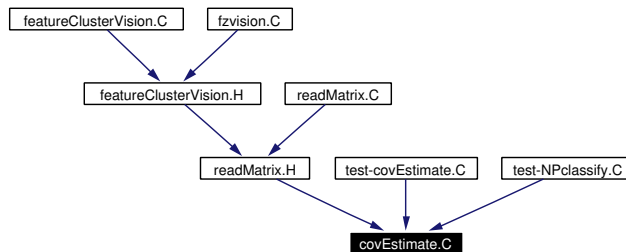
```
#include "Point2D.H"
```

```
#include <fstream>
```

Include dependency graph for covEstimate.C:



This graph shows which files directly or indirectly include this file:



### 11.96.1 Detailed Description

quick estimator for covariance matrix

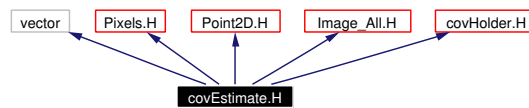
Definition in file [covEstimate.C](#).



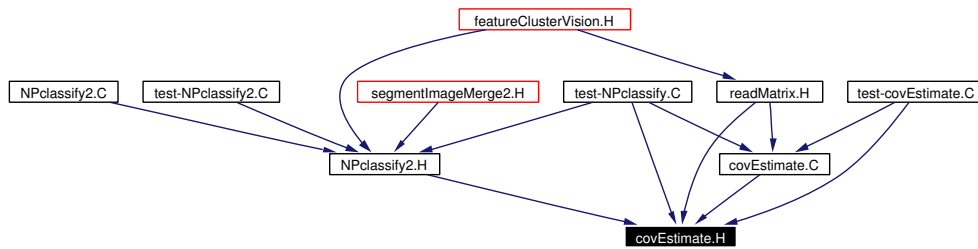
## 11.97 covEstimate.H File Reference

```
#include <vector>
#include "Pixels.H"
#include "Point2D.H"
#include "Image_All.H"
#include "covHolder.H"
```

Include dependency graph for covEstimate.H:



This graph shows which files directly or indirectly include this file:



### Classes

- class **covEstimate**

#### 11.97.1 Detailed Description

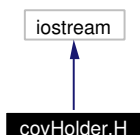
quick estimator for coveriance matrix

Definition in file [covEstimate.H](#).

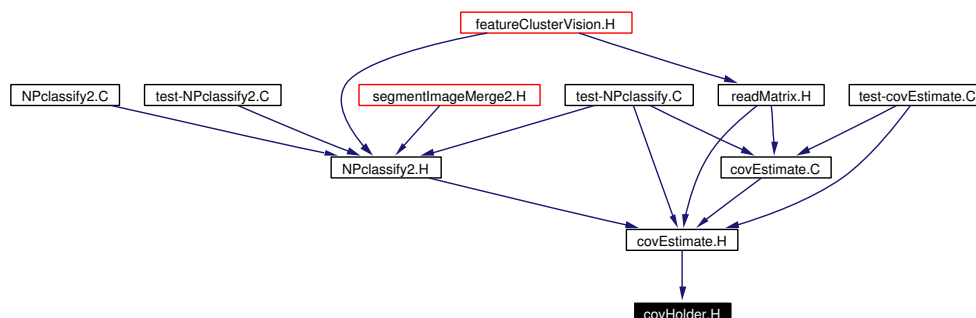
## 11.98 covHolder.H File Reference

```
#include <iostream>
```

Include dependency graph for covHolder.H:



This graph shows which files directly or indirectly include this file:



### Classes

- class **covHolder**

#### 11.98.1 Detailed Description

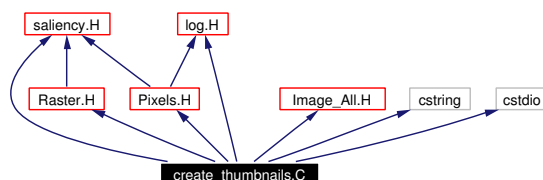
holds properaties of objects

Definition in file [covHolder.H](#).

## 11.99 create\_thumbnails.C File Reference

```
#include "saliency.H"
#include "log.H"
#include "Image_All.H"
#include "Raster.H"
#include <cstring>
#include <stdio>
#include "Pixels.H"
```

Include dependency graph for create\_thumbnails.C:



### Defines

- `#define TSIZEX 64`
- `#define TBORDER 2`
- `#define TSIZEY (TSIZEX + TBORDER)`
- `#define TNAME "thumbnails"`

### Functions

- `int main (int argc, char **argv)`

#### 11.99.1 Detailed Description

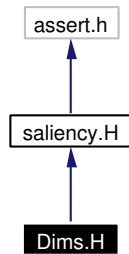
Create small images from originals

Definition in file [create\\_thumbnails.C](#).

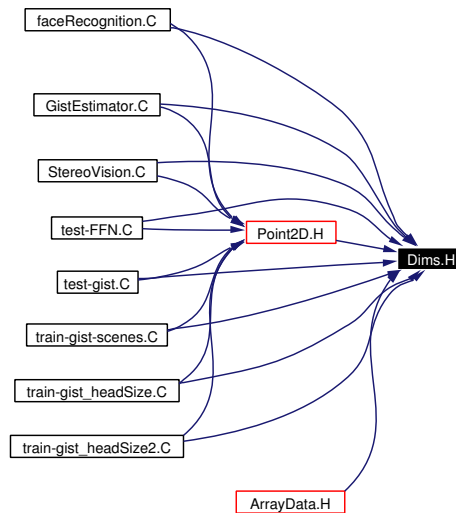
## 11.100 Dims.H File Reference

```
#include "saliency.H"
```

Include dependency graph for Dims.H:



This graph shows which files directly or indirectly include this file:



## Classes

- class [Dims](#)

*A simple struct to hold a pair of width/height dimensions.*

### 11.100.1 Detailed Description

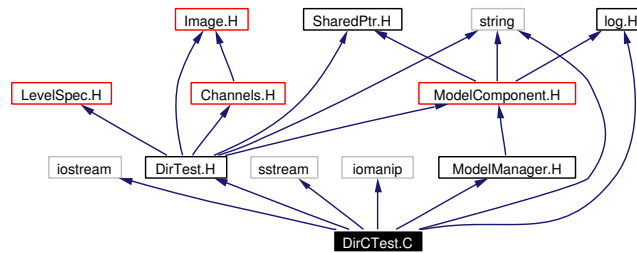
A simple struct to hold a pair of width/height dimensions.

Definition in file [Dims.H](#).

## 11.101 DirCTest.C File Reference

```
#include <iostream>
#include <string>
#include <sstream>
#include <iomanip>
#include "DirTest.H"
#include "log.H"
#include "ModelManager.H"
```

Include dependency graph for DirCTest.C:



### Classes

- struct **Args**

### Functions

- int **main** (int argc, char \*\*argv)

#### 11.101.1 Detailed Description

This program is for testing the [DirectionChannel](#). It takes as input from the command line in the form:

```
DirCTest [OPTIONS] imageBaseName numFrames dir
```

This program uses the helper class [DirTest](#) in [DirTest.HC](#) to do most of the work.

The options determine which maps are displayed, whether the images are normalized by [XWinManaged](#) before they are displayed and the formatting of the statistics output.

The valid options are: -rs -rm -gps -gpm -ss -sm -cs -gm -dm -norm -compact

r = Reichardt gp = Gabor pyramid s = Submap (feature) c = conspicuity (both from gabor and direction) g = gabor d = direction

The suffixes are:

s = statistics (min, max, avg) m = maps

Others:

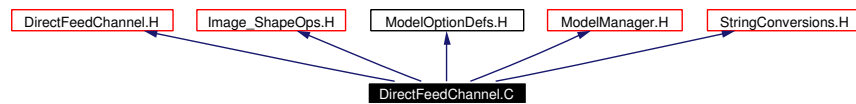
norm - normalize images compact - display all the statistics for each map without any cr/lf between them (less room on screen)

Definition in file [DirCTest.C](#).

## 11.102 DirectFeedChannel.C File Reference

```
#include "DirectFeedChannel.H"  
#include "Image_ShapeOps.H"  
#include "ModelOptionDefs.H"  
#include "ModelManager.H"  
#include "StringConversions.H"
```

Include dependency graph for DirectFeedChannel.C:



### 11.102.1 Detailed Description

the source for [DirectFeedChannel](#) and [MultiDirectFeedChannel](#)

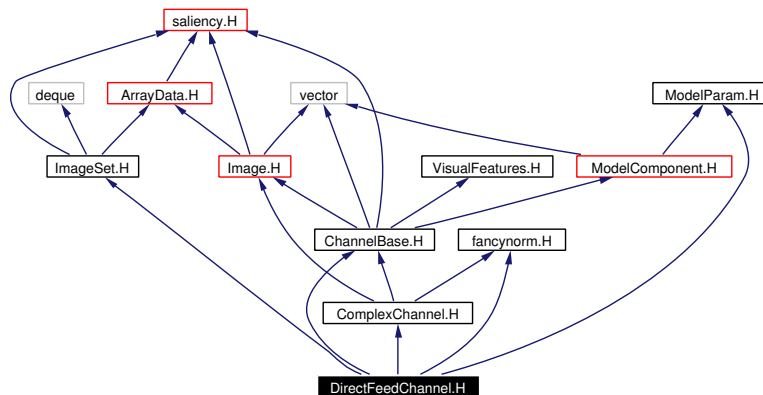
Definition in file [DirectFeedChannel.C](#).



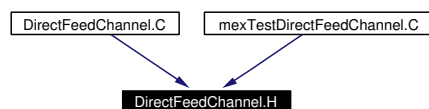
## 11.103 DirectFeedChannel.H File Reference

```
#include "ChannelBase.H"
#include "ComplexChannel.H"
#include "ImageSet.H"
#include "fancynorm.H"
#include "ModelParam.H"
```

Include dependency graph for DirectFeedChannel.H:



This graph shows which files directly or indirectly include this file:



## Classes

- class [DirectFeedChannel](#)  
*Direct Feed Channel.*
- class [MultiDirectFeedChannel](#)  
*Multi Direct Feed Channel.*

## Typedefs

- typedef [std::vector](#)< [ImageSet](#)< float > > [PyrVec](#)

*A vector of ImageSets - to be used as input to [MultiDirectFeedChannel](#).*

### 11.103.1 Detailed Description

the source for [DirectFeedChannel](#) and [MultiDirectFeedChannel](#)

Definition in file [DirectFeedChannel.H](#).

### 11.103.2 Typedef Documentation

#### 11.103.2.1 typedef [std::vector](#)< [ImageSet](#)<float> > [PyrVec](#)

A vector of ImageSets - to be used as input to [MultiDirectFeedChannel](#).

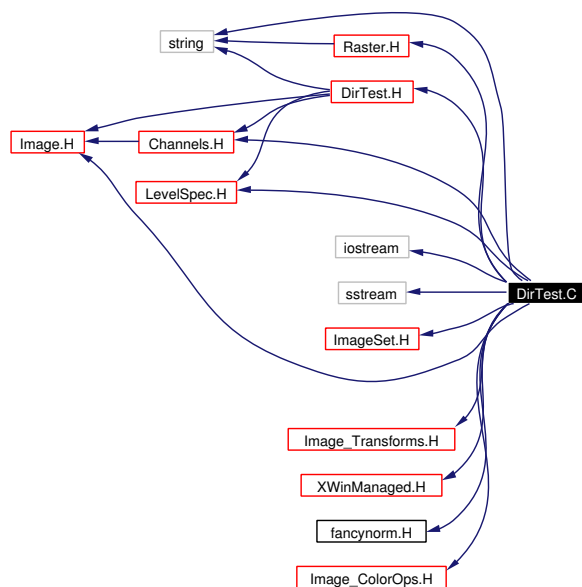
Definition at line 216 of file [DirectFeedChannel.H](#).

Referenced by [MultiDirectFeedChannel::inputPyramidVector\(\)](#).

## 11.104 DirTest.C File Reference

```
#include "DirTest.H"  
#include <iostream>  
#include <string>  
#include <sstream>  
#include "Image.H"  
#include "ImageSet.H"  
#include "Raster.H"  
#include "LevelSpec.H"  
#include "Channels.H"  
#include "Image_Transforms.H"  
#include "XWinManaged.H"  
#include "fancynorm.H"  
#include "Image_ColorOps.H"
```

Include dependency graph for DirTest.C:



## Defines

- `#define Q_LEN 2`
- `#define SML 4`
- `#define DELTA_MIN 3`
- `#define DELTA_MAX 4`
- `#define LEVEL_MIN 2`
- `#define LEVEL_MAX 4`
- `#define CHANOPTS SingleChannel::NO_OPTS`

### 11.104.1 Detailed Description

defines methods for class [DirTest](#)

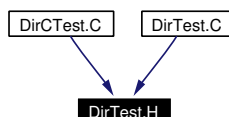
Definition in file [DirTest.C](#).

## 11.105 DirTest.H File Reference

```
#include <string>
#include "Image.H"
#include "LevelSpec.H"
#include "SharedPtr.H"
#include "Channels.H"
#include "ModelComponent.H"
```

Include dependency graph for DirTest.H:

This graph shows which files directly or indirectly include this file:



### Classes

- class [DirTest](#)  
*Class for testing [DirectionChannel](#).*

#### 11.105.1 Detailed Description

defines class [DirTest](#), used for testing the class

Definition in file [DirTest.H](#).

## 11.106 dummySTL.H File Reference

### Namespaces

- namespace [std](#)

### 11.106.1 Detailed Description

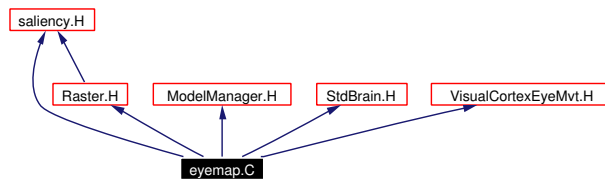
A dummy STL definition for doxygen **DO NOT INCLUDE THIS FILE!!!**

Definition in file [dummySTL.H](#).

## 11.107 eyemap.C File Reference

```
#include "saliency.H"
#include "ModelManager.H"
#include "Raster.H"
#include "StdBrain.H"
#include "VisualCortexEyeMvt.H"
```

Include dependency graph for eyemap.C:



### Defines

- `#define` `MAXSTIME` 30.0  
*Maximum run time in seconds.*

### Functions

- `int` `main` (`const int argc`, `const char **argv`)  
*Build a human eye movement map.*

#### 11.107.1 Detailed Description

create a human eye density map from eye tracking traces

Definition in file [eyemap.C](#).

#### 11.107.2 Define Documentation

##### 11.107.2.1 `#define` `MAXSTIME` 30.0

Maximum run time in seconds.

Definition at line 45 of file eyemap.C.

Referenced by main().

### 11.107.3 Function Documentation

#### 11.107.3.1 `int main (const int argc, const char ** argv)`

Build a human eye movement map.

Definition at line 48 of file eyemap.C.

References `SharedPtr< T >::dynCastFrom()`, `main()`, `MAXSTIME`, `MYLOGVERB`, `Raster::ReadRGB()`, `uint`, and `Raster::WriteFloat()`.



## 11.108 ezvision.C File Reference

```
#include "saliency.H"
#include "ModelManager.H"
#include "StdBrain.H"
#include "FrameSeries.H"
#include "ModelOptionAliases.H"
```

Include dependency graph for ezvision.C:

### Functions

- int **main** (const int argc, const char \*\*argv)

### 11.108.1 Detailed Description

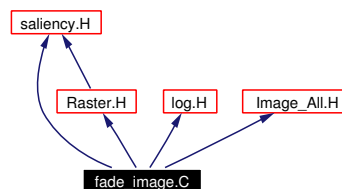
simplified version of [vision.C](#)

Definition in file [ezvision.C](#).

## 11.109 fade\_image.C File Reference

```
#include "saliency.H"  
#include "log.H"  
#include "Image_All.H"  
#include "Raster.H"
```

Include dependency graph for fade\_image.C:



### Functions

- int **main** (int argc, char \*\*argv)

### 11.109.1 Detailed Description

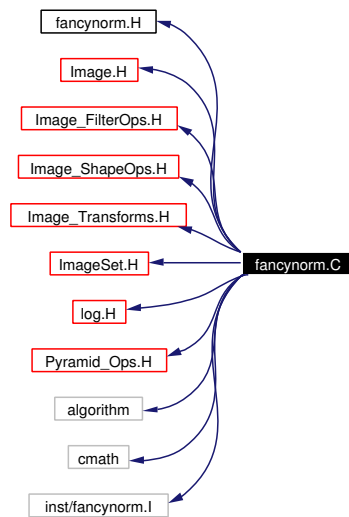
: video fade effects

Definition in file [fade\\_image.C](#).

## 11.110 fancynorm.C File Reference

```
#include "fancynorm.H"
#include "Image.H"
#include "Image_FilterOps.H"
#include "Image_ShapeOps.H"
#include "Image_Transforms.H"
#include "ImageSet.H"
#include "log.H"
#include "Pyramid_Ops.H"
#include <algorithm>
#include <cmath>
#include "inst/fancynorm.I"
```

Include dependency graph for fancynorm.C:



## Functions

- `template<class T> Image< T > maxNormalize (const Image< T > &src, const T mi, const T ma, const MaxNormType normtyp, int nbiter, const Image< float > *lrexcit)`

*Generic implementation (select algorithm with normtyp).*

- `template<class T> Image< T > maxNormalizeNone` (const `Image< T > &src`, const `T mi`, const `T ma`)  
*Specialized implementation (corresponds to VCXNORM\_NONE).*
- `template<class T> Image< T > maxNormalizeStd` (const `Image< T > &src`, const `T mi`, const `T ma`)  
*Specialized implementation (corresponds to VCXNORM\_MAXNORM).*
- `template<class T> Image< T > maxNormalizeFancyFast` (const `Image< T > &src`, const `T mi`, const `T ma`, const `int nbiter`)  
*Specialized implementation (corresponds to VCXNORM\_FANCYFAST).*
- `template<class T> Image< T > maxNormalizeFancy` (const `Image< T > &src`, const `T mi`, const `T ma`, const `int nbiter`, const `Image< float > *lrexcit`)  
*Specialized implementation (corresponds to VCXNORM\_FANCY).*
- `template<class T> Image< T > maxNormalizeFancyWeak` (const `Image< T > &src`, const `T mi`, const `T ma`)  
*Weak version of FancyOne.*
- `template<class T> Image< T > maxNormalizeFancyLandmark` (const `Image< T > &src`, const `T mi`, const `T ma`, const `int nbiter`)  
*Specialized implementation (corresponds to VCXNORM\_FANCY).*
- `template<class T> Image< T > maxNormalizeLandmark` (const `Image< T > &src`, const `T mi`, const `T ma`)  
*Specialized implementation (corresponds to VCXNORM\_LANDMARK).*
- `template<class T> int findPeaks` (const `Image< T > &src`, const `T mi`, const `T ma`, double `&sum_activity`)  
*to find number of peaks in the image (used in landmark detection)*
- `template<class T> float goodness_map` (const `Image< T > &src`)  
*to find the goodness of a map (used in landmark detection)*

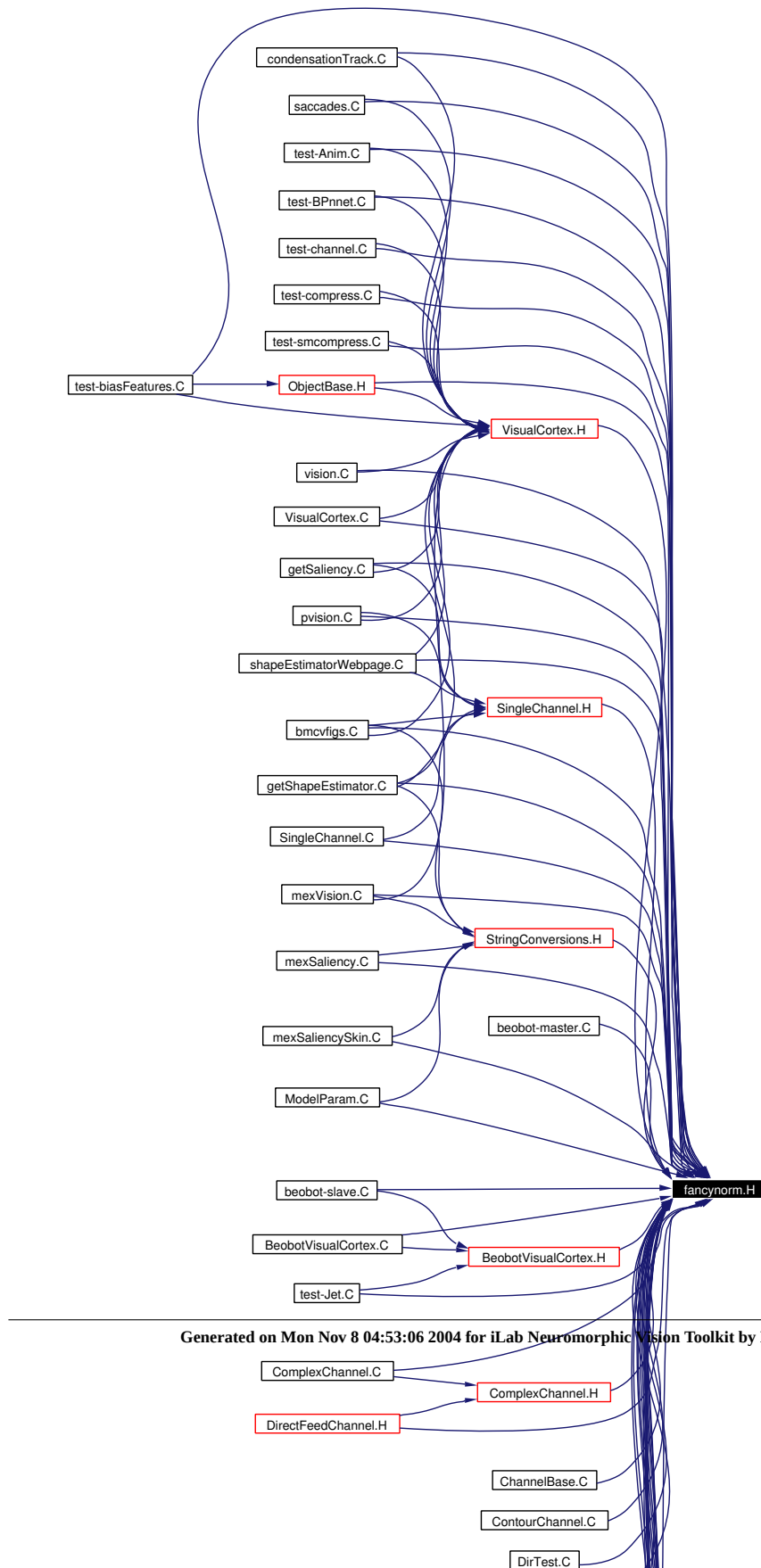
### 11.110.1 Detailed Description

Intrafeature competition with `maxNormalize()`.

Definition in file `fancynorm.C`.

## 11.111 fancynorm.H File Reference

This graph shows which files directly or indirectly include this file:



## Types of normalization (normtyp)

- #define NBMAXNORMTYPES 10
- #define VCXNORM\_DEFAULT VCXNORM\_FANCY
- enum MaxNormType {  
    VCXNORM\_NONE = 0, VCXNORM\_MAXNORM = 1, VCXNORM\_FANCY = 2, VCXNORM\_FANCYFAST = 3,  
    VCXNORM\_FANCYONE = 4, VCXNORM\_FANCYLANDMARK = 5, VCXNORM\_LANDMARK = 6, VCXNORM\_FANCYWEAK = 7,  
    VCXNORM\_IGNORE = 8, VCXNORM\_SURPRISE = 9 }  
• const char \* maxNormTypeName (const MaxNormType m)  
    *Get a name in clear for a given type.*

## Additional parameters for fancynorm versions of the algorithm.

- const int FANCYITER = 5  
    *default number of iterations*
- const double FANCYESIG = 2  
    *excitatory sigma as of image size*
- const double FANCYISIG = 25  
    *inhibitory sigma as of image size*
- const double FANCYCOEX = 0.5  
    *excitatory coefficient (strength)*
- const double FANCYCOIN = 1.5  
    *inhibitory coefficient (strength)*
- const double FANCYINHI = 2.0  
    *strength of global inhibition*
- const double FANCYG = 2.1  
    *for sigmoid normalization*
- const double FANCYH = 2.0  
    *for sigmoid normalization*
- const double FANCYS = 1.0  
    *for sigmoid normalization*

- const int `LRLEVEL` = 2  
*for tuned long-range excitation:*

### Standard min/max bounds for `maxNormalize()`.

- const float `MAXNORMMAX` = 10.0f  
*upper bound for `maxNormalize()`*
- const float `MAXNORMMIN` = 0.0f  
*lower bound for `maxNormalize()`*
- const float `MAXNORMLANDMARK` = 255.0f  
*upper bound for `maxNormalizeFancyLandmark()`*

### Functions

- template<class T> `Image< T > maxNormalize` (const `Image< T > &src`, const T mi, const T ma, const `MaxNormType` normtyp=`VCXNORM_DEFAULT`, int nbiter=`FANCYITER`, const `Image< float > *lrexcit=0`)  
*Generic implementation (select algorithm with normtyp).*
- template<class T> `Image< T > maxNormalizeNone` (const `Image< T > &src`, const T mi, const T ma)  
*Specialized implementation (corresponds to `VCXNORM_NONE`).*
- template<class T> `Image< T > maxNormalizeStd` (const `Image< T > &src`, const T mi, const T ma)  
*Specialized implementation (corresponds to `VCXNORM_MAXNORM`).*
- template<class T> `Image< T > maxNormalizeFancyFast` (const `Image< T > &src`, const T mi, const T ma, const int nbiter=`FANCYITER`)  
*Specialized implementation (corresponds to `VCXNORM_FANCYFAST`).*
- template<class T> `Image< T > maxNormalizeFancy` (const `Image< T > &src`, const T mi, const T ma, const int nbiter=`FANCYITER`, const `Image< float > *lrexcit=0`)  
*Specialized implementation (corresponds to `VCXNORM_FANCY`).*



- `template<class T> Image< T > maxNormalizeFancyWeak (const Image< T > &src, const T mi, const T ma)`  
*Weak version of FancyOne.*
- `template<class T> Image< T > maxNormalizeFancyLandmark (const Image< T > &src, const T mi, const T ma, const int nbiter=FANCYITER)`  
*Specialized implementation (corresponds to VCXNORM\_FANCY).*
- `template<class T> Image< T > maxNormalizeLandmark (const Image< T > &src, const T mi, const T ma)`  
*Specialized implementation (corresponds to VCXNORM\_LANDMARK).*
- `template<class T> int findPeaks (const Image< T > &src, const T mi, const T ma, double &sum)`  
*to find number of peaks in the image (used in landmark detection)*
- `template<class T> float goodness_map (const Image< T > &src)`  
*to find the goodness of a map (used in landmark detection)*

### 11.111.1 Detailed Description

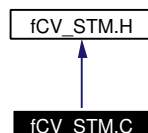
Intrafeature competition with `maxNormalize()`.

Definition in file `fancynorm.H`.

## 11.112 fCV\_STM.C File Reference

```
#include "fCV_STM.H"
```

Include dependency graph for fCV\_STM.C:



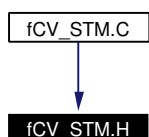
### 11.112.1 Detailed Description

Short Term Memory for Feature Cluster Vision

Definition in file [fCV\\_STM.C](#).

## 11.113 fCV\_STM.H File Reference

This graph shows which files directly or indirectly include this file:



### Classes

- class **fCV\_STM**

#### 11.113.1 Detailed Description

Short Term Memory for Feature Cluster Vision

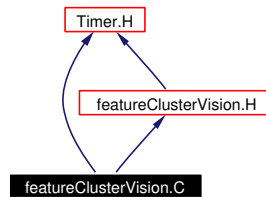
Definition in file [fCV\\_STM.H](#).

## 11.114 featureClusterVision.C File Reference

```
#include "featureClusterVision.H"
```

```
#include "Timer.H"
```

Include dependency graph for featureClusterVision.C:



### Defines

- `#define OVER 5`
- `#define COVSIZE 250`
- `#define DATABOUND 10000`

### 11.114.1 Detailed Description

Test the nonparametric classifier

Definition in file [featureClusterVision.C](#).

## 11.115 FFN.C File Reference

```
#include "FFN.H"
```

```
#include "log.H"
```

Include dependency graph for FFN.C:

### 11.115.1 Detailed Description

Feed Forward Network for use to train any mapping

Definition in file [FFN.C](#).

## 11.116 FFN.H File Reference

```
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include <time.h>
```

Include dependency graph for FFN.H:

This graph shows which files directly or indirectly include this file:

### Classes

- class [FeedForwardNetwork](#)  
*A Feed Forward Network.*

### Defines

- #define **START\_RANGE** 1.0

#### 11.116.1 Detailed Description

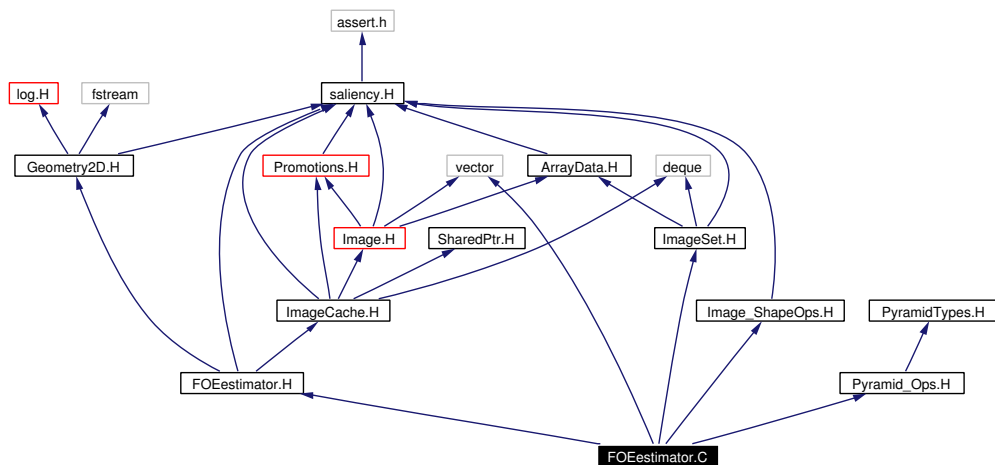
Feed Forward Network for use to train any mapping

Definition in file [FFN.H](#).

## 11.117 FOEstimator.C File Reference

```
#include "FOEstimator.H"
#include "ImageSet.H"
#include "Pyramid_Ops.H"
#include "Image_ShapeOps.H"
#include <vector>
```

Include dependency graph for FOEstimator.C:



### 11.117.1 Detailed Description

a class for estimating the focus of expansion in a video

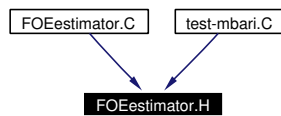
Definition in file [FOEstimator.C](#).

## 11.118 FOEstimator.H File Reference

```
#include "saliency.H"  
#include "ImageCache.H"  
#include "Geometry2D.H"
```

Include dependency graph for FOEstimator.H:

This graph shows which files directly or indirectly include this file:



### Classes

- class [FOEstimator](#)  
*compute the focus of expansion (FOE) from the pixel-based optical flow*

### 11.118.1 Detailed Description

a class for estimating the focus of expansion in a video

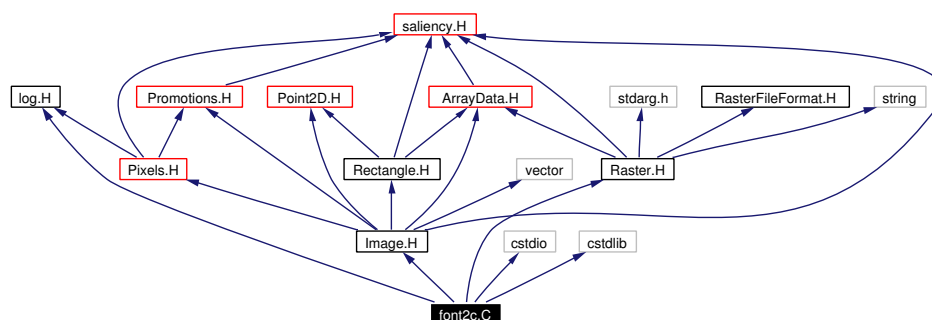
Definition in file [FOEstimator.H](#).



## 11.119 font2c.C File Reference

```
#include "Image.H"
#include "log.H"
#include "Raster.H"
#include <stdio>
#include <stdlib>
```

Include dependency graph for font2c.C:



### Defines

- `#define NBCHARS 95`

### Functions

- `int main (int argc, char **argv)`

#### 11.119.1 Detailed Description

Takes a .pgm image with ASCII chars written in it, and transforms it into C source code Definition in file [font2c.C](#).

## 11.119.2 Function Documentation

### 11.119.2.1 `int main (int argc, char ** argv)`

[font2c.C](#) – takes a .pgm image with ASCII chars written in it, and transforms it into C source code

Definition at line 56 of file font2c.C.

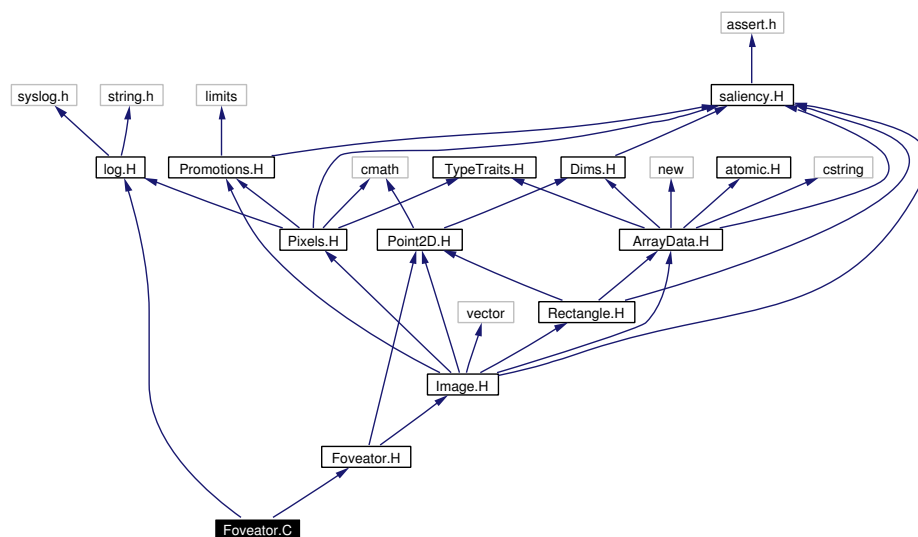
References `Image< T >::getHeight()`, `Image< T >::getVal()`, `Image< T >::getWidth()`, `main()`, and `Raster::ReadGray()`.

## 11.120 Foveator.C File Reference

```
#include "log.H"
```

```
#include "Foveator.H"
```

Include dependency graph for Foveator.C:



### 11.120.1 Detailed Description

An abstract class for space-variant image processing

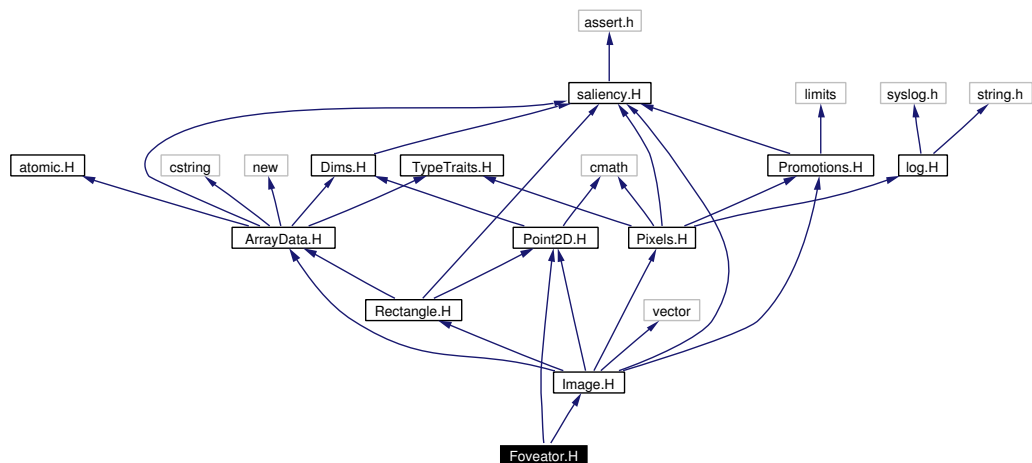
Definition in file [Foveator.C](#).

## 11.121 Foveator.H File Reference

```
#include "Image.H"
```

```
#include "Point2D.H"
```

Include dependency graph for Foveator.H:



This graph shows which files directly or indirectly include this file:

### Classes

- class [Foveator](#)  
*An abstract class for space-variant image processing.*

### Defines

- #define **MAX**(x, y) ( ( (x) > (y) ) ? (x) : (y) )
- #define **MIN**(x, y) ( ( (x) < (y) ) ? (x) : (y) )

### 11.121.1 Detailed Description

An abstract class for space-variant image processing

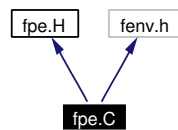
Definition in file [Foveator.H](#).

## 11.122 fpe.C File Reference

```
#include "fpe.H"
```

```
#include <fenv.h>
```

Include dependency graph for fpe.C:



### Functions

- void [fpExceptionsOn](#) ()  
*Turn floating-point exceptions on.*
- void [fpExceptionsOff](#) ()  
*Turn floating-point exceptions off.*
- void [fpExceptionsLock](#) ()  
*Lock floating-point exceptions in their current state.*
- void [fpExceptionsUnlock](#) ()  
*Unlock floating-point exceptions.*
- bool [fpIsEnabled](#) ()  
*Check whether floating-point exceptions are on.*
- bool [fpIsLocked](#) ()  
*Check whether floating-point exceptions are locked in their current state.*

### Variables

- bool **isEnabled** = false
- bool **isLocked** = false

### 11.122.1 Detailed Description

floating-point exceptions

Definition in file [fpe.C](#).

### 11.122.2 Function Documentation

#### 11.122.2.1 void fpExceptionsLock ()

Lock floating-point exceptions in their current state.

That is, if FPEs are on now, they will stay locked on even if someone calls [fpExceptionsOff\(\)](#); likewise, if FPEs are off now, they will stay locked off even if someone calls [fpExceptionsOn\(\)](#).

Definition at line 70 of file [fpe.C](#).

Referenced by [MexModelManager::MexModelManager\(\)](#).

#### 11.122.2.2 void fpExceptionsOff ()

Turn floating-point exceptions off.

Definition at line 59 of file [fpe.C](#).

Referenced by [MexModelManager::MexModelManager\(\)](#), [ModelManager::setModelParamString\(\)](#), and [ToolkitInit::ToolkitInit\(\)](#).

#### 11.122.2.3 void fpExceptionsOn ()

Turn floating-point exceptions on.

Definition at line 48 of file [fpe.C](#).

Referenced by [ModelManager::setModelParamString\(\)](#), and [ToolkitInit::ToolkitInit\(\)](#).

#### 11.122.2.4 void fpExceptionsUnlock ()

Unlock floating-point exceptions.

This does not turn FPEs on or off, but does make it possible for the state to be changed again via [fpExceptionsOn\(\)](#) or [fpExceptionsOff\(\)](#).

Definition at line 76 of file [fpe.C](#).

Referenced by [MexModelManager::MexModelManager\(\)](#).

**11.122.2.5 bool fpIsEnabled ()**

Check whether floating-point exceptions are on.

Definition at line 82 of file fpe.C.

Referenced by ModelManager::ModelManager().

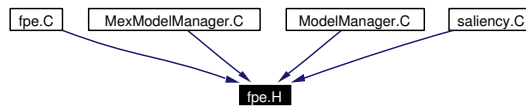
**11.122.2.6 bool fpIsLocked ()**

Check whether floating-point exceptions are locked in their current state.

Definition at line 88 of file fpe.C.

## 11.123 fpe.H File Reference

This graph shows which files directly or indirectly include this file:



### Functions

- void [fpExceptionsOn](#) ()  
*Turn floating-point exceptions on.*
- void [fpExceptionsOff](#) ()  
*Turn floating-point exceptions off.*
- void [fpExceptionsLock](#) ()  
*Lock floating-point exceptions in their current state.*
- void [fpExceptionsUnlock](#) ()  
*Unlock floating-point exceptions.*
- bool [fpIsEnabled](#) ()  
*Check whether floating-point exceptions are on.*
- bool [fpIsLocked](#) ()  
*Check whether floating-point exceptions are locked in their current state.*

### 11.123.1 Detailed Description

floating-point exceptions

Definition in file [fpe.H](#).

### 11.123.2 Function Documentation

#### 11.123.2.1 void [fpExceptionsLock](#) ()

Lock floating-point exceptions in their current state.



That is, if FPEs are on now, they will stay locked on even if someone calls [fpExceptionsOff\(\)](#); likewise, if FPEs are off now, they will stay locked off even if someone calls [fpExceptionsOn\(\)](#).

Definition at line 70 of file fpe.C.

Referenced by `MexModelManager::MexModelManager()`.

#### 11.123.2.2 void fpExceptionsOff ()

Turn floating-point exceptions off.

Definition at line 59 of file fpe.C.

Referenced by `MexModelManager::MexModelManager()`, `ModelManager::setModelParamString()`, and `ToolkitInit::ToolkitInit()`.

#### 11.123.2.3 void fpExceptionsOn ()

Turn floating-point exceptions on.

Definition at line 48 of file fpe.C.

Referenced by `ModelManager::setModelParamString()`, and `ToolkitInit::ToolkitInit()`.

#### 11.123.2.4 void fpExceptionsUnlock ()

Unlock floating-point exceptions.

This does not turn FPEs on or off, but does make it possible for the state to be changed again via [fpExceptionsOn\(\)](#) or [fpExceptionsOff\(\)](#).

Definition at line 76 of file fpe.C.

Referenced by `MexModelManager::MexModelManager()`.

#### 11.123.2.5 bool fpIsEnabled ()

Check whether floating-point exceptions are on.

Definition at line 82 of file fpe.C.

Referenced by `ModelManager::ModelManager()`.

#### 11.123.2.6 bool fpIsLocked ()

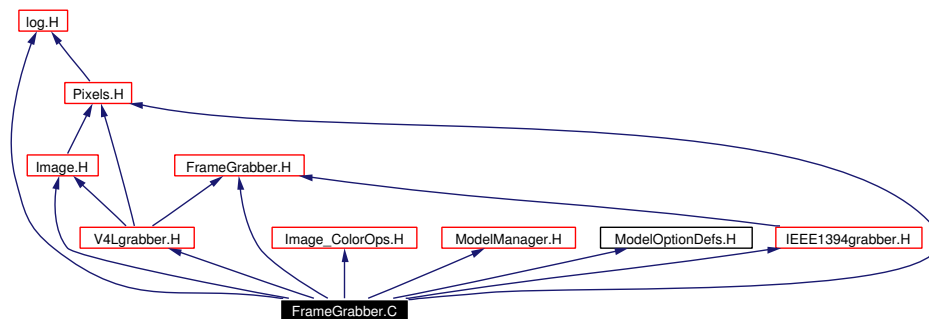
Check whether floating-point exceptions are locked in their current state.

Definition at line 88 of file fpe.C.

## 11.124 FrameGrabber.C File Reference

```
#include "FrameGrabber.H"  
#include "log.H"  
#include "Image.H"  
#include "Pixels.H"  
#include "Image_ColorOps.H"  
#include "ModelManager.H"  
#include "ModelOptionDefs.H"  
#include "V4Lgrabber.H"  
#include "IEEE1394grabber.H"
```

Include dependency graph for FrameGrabber.C:



### 11.124.1 Detailed Description

Base class for a frame grabber device

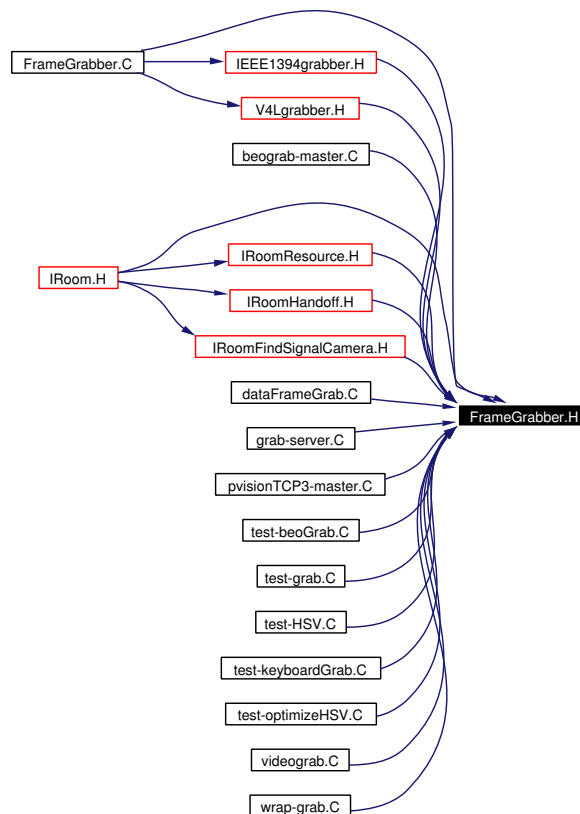
Definition in file [FrameGrabber.C](#).

## 11.125 FrameGrabber.H File Reference

```
#include "saliency.H"
#include "FrameGrabberMode.H"
#include "ArrayData.H"
#include "ModelComponent.H"
```

Include dependency graph for FrameGrabber.H:

This graph shows which files directly or indirectly include this file:



## Classes

- class [FrameGrabber](#)

*Base class for a frame grabber device.*

- class [FrameGrabberConfigurator](#)

*This class is for run-time selection of a frame grabber driver.*

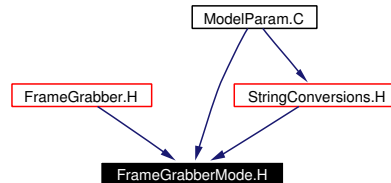
### 11.125.1 Detailed Description

Base class for a frame grabber device

Definition in file [FrameGrabber.H](#).

## 11.126 FrameGrabberMode.H File Reference

This graph shows which files directly or indirectly include this file:



### Defines

- `#define` [NBFRAMEGRABBERMODES](#) 19  
*number of known frame grabber modes*

### Enumerations

- `enum` [FrameGrabberMode](#) {  
**GMGREY** = 0, **GMRRAW** = 1, **GMRGB555** = 2, **GMRGB565** = 3,  
**GMRGB24** = 4, **GMRGB32** = 5, **GMUYUV** = 6, **GMUYVY** = 7,  
**GMYUV444** = 8, **GMYUV422** = 9, **GMYUV411** = 10, **GMYUV420** = 11,  
**GMYUV410** = 12, **GMYUV444P** = 13, **GMYUV422P** = 14, **GMYUV411P** = 15,  
**GMYUV420P** = 16, **GMYUV410P** = 17, **GMAUTO** }  
*[FrameGrabber](#) grab modes. Not all may be supported by various hardware.*

### Functions

- `const char *` [frameGrabberModeName](#) (`const` [FrameGrabberMode](#) `val`)  
*Text descriptions of the various known grabmodes:.*

#### 11.126.1 Detailed Description

definitions of possible image grabbing modes

Definition in file [FrameGrabberMode.H](#).

## 11.126.2 Define Documentation

### 11.126.2.1 `#define NBFRAMEGRABBERMODES 19`

number of known frame grabber modes

Definition at line 68 of file FrameGrabberMode.H.

Referenced by `convertFromString()`.

## 11.126.3 Enumeration Type Documentation

### 11.126.3.1 `enum FrameGrabberMode`

`FrameGrabber` grab modes. Not all may be supported by various hardware.

See [FrameGrabber.H](#) for functions using those modes. If you add modes here, be sure to update the string descriptions in the [frameGrabberModeName\(\)](#) function below.

Definition at line 45 of file FrameGrabberMode.H.

Referenced by `convertFromString()`, and `FrameGrabber::FrameGrabber()`.

## 11.126.4 Function Documentation

### 11.126.4.1 `const char* frameGrabberModeName (const FrameGrabberMode val) [inline]`

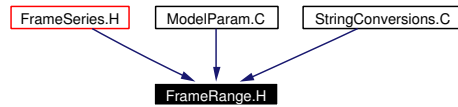
Text descriptions of the various known grabmodes:.

Definition at line 71 of file FrameGrabberMode.H.

Referenced by `convertFromString()`, `convertToString()`, and `V4Lgrabber-Light::V4LgrabberLight()`.

## 11.127 FrameRange.H File Reference

This graph shows which files directly or indirectly include this file:



### Classes

- class [FrameRange](#)  
*Specification of a range of frames with given inter-frame delay (in ms).*

### 11.127.1 Detailed Description

A range of frames at a given inter-frame delay

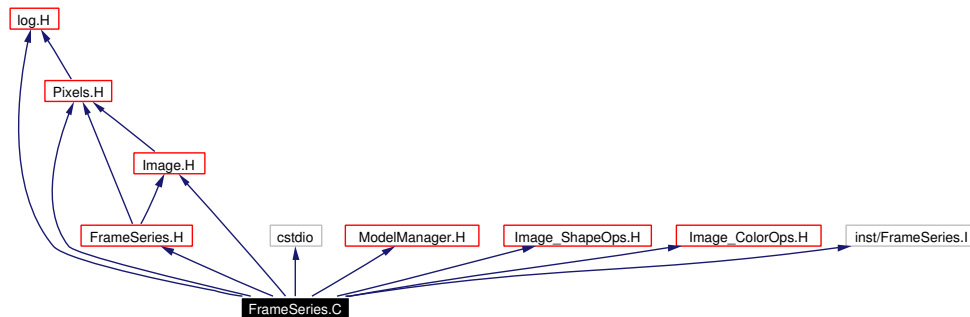
Definition in file [FrameRange.H](#).



## 11.128 FrameSeries.C File Reference

```
#include "FrameSeries.H"  
#include "log.H"  
#include <cstdio>  
#include "Pixels.H"  
#include "Image.H"  
#include "ModelManager.H"  
#include "Image_ShapeOps.H"  
#include "Image_ColorOps.H"  
#include "inst/FrameSeries.I"
```

Include dependency graph for FrameSeries.C:



### Defines

- `#define MAXDISP 20`
- `#define INST_CLASS FrameSeries::`

### 11.128.1 Detailed Description

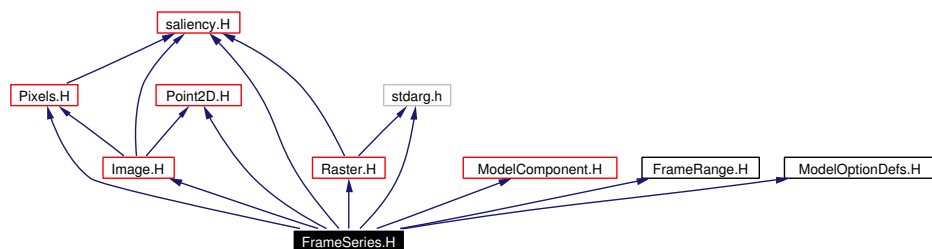
a series of frames

Definition in file [FrameSeries.C](#).

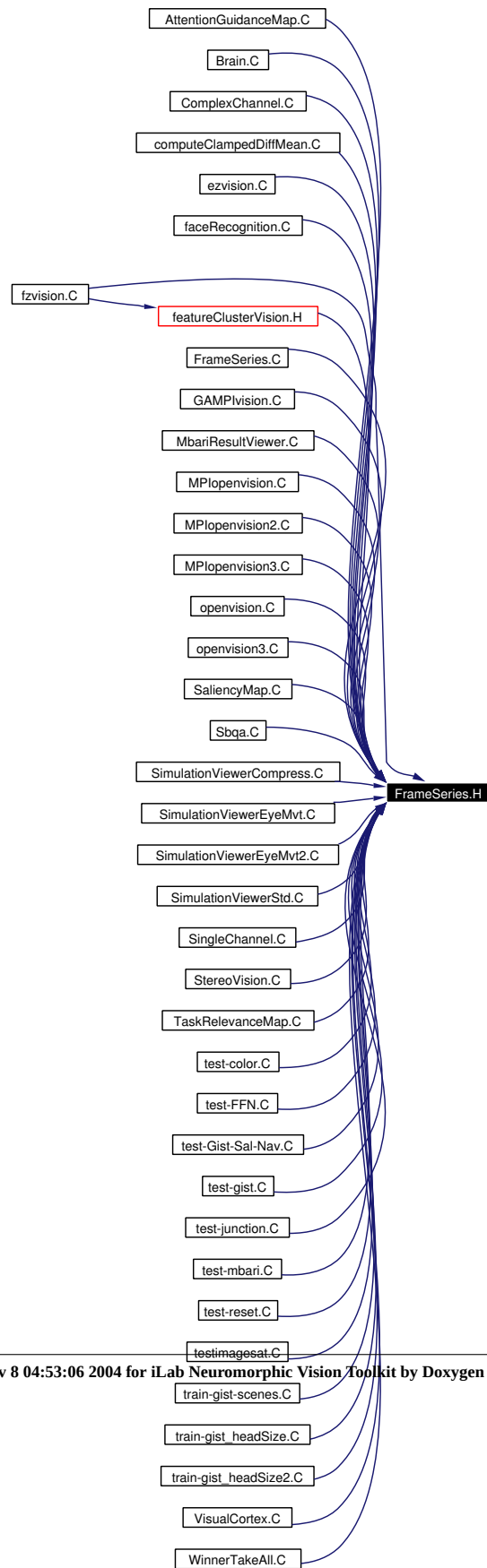
## 11.129 FrameSeries.H File Reference

```
#include "saliency.H"  
#include "ModelComponent.H"  
#include "Raster.H"  
#include <stdarg.h>  
#include "FrameRange.H"  
#include "Point2D.H"  
#include "Pixels.H"  
#include "Image.H"  
#include "ModelOptionDefs.H"
```

Include dependency graph for FrameSeries.H:



This graph shows which files directly or indirectly include this file:



## Classes

- class [FrameSeries](#)  
*FrameSeries manages a series of input or output frames.*
- class [InputFrameSeries](#)  
*A [FrameSeries](#) with additional facilities for reading input frames.*
- class [OutputFrameSeries](#)  
*A [FrameSeries](#) with additional facilities for writing output frames.*

### 11.129.1 Detailed Description

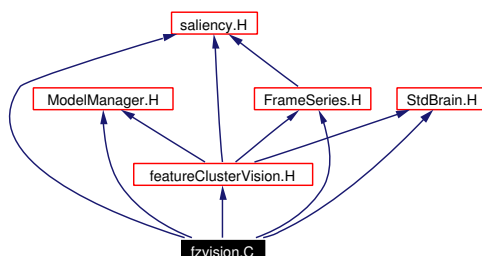
a series of frames

Definition in file [FrameSeries.H](#).

## 11.130 fzvision.C File Reference

```
#include "saliency.H"
#include "ModelManager.H"
#include "StdBrain.H"
#include "FrameSeries.H"
#include "featureClusterVision.H"
```

Include dependency graph for fzvision.C:



### Functions

- int **main** (const int argc, const char \*\*argv)

#### 11.130.1 Detailed Description

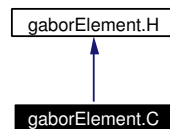
simplified version of [vision.C](#) with feature analysis

Definition in file [fzvision.C](#).

## 11.131 gaborElement.C File Reference

```
#include "gaborElement.H"
```

Include dependency graph for gaborElement.C:



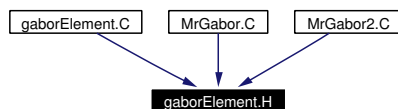
### 11.131.1 Detailed Description

FIXME

Definition in file [gaborElement.C](#).

## 11.132 gaborElement.H File Reference

This graph shows which files directly or indirectly include this file:



### Classes

- class **gaborElement**

#### 11.132.1 Detailed Description

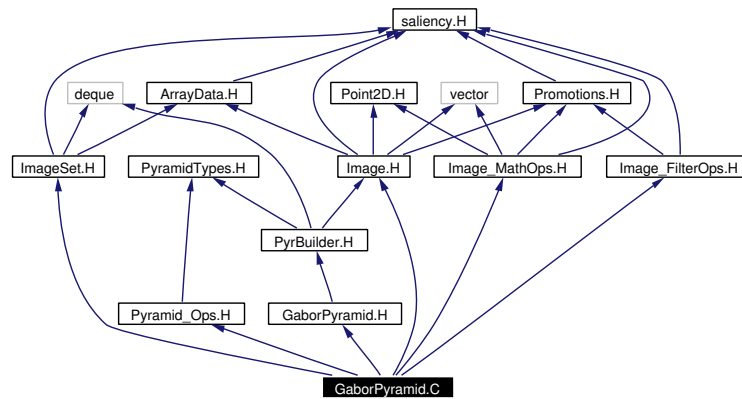
A Gabor element

Definition in file [gaborElement.H](#).

## 11.133 GaborPyramid.C File Reference

```
#include "GaborPyramid.H"
#include "Image.H"
#include "Image_FilterOps.H"
#include "Image_MathOps.H"
#include "ImageSet.H"
#include "Pyramid_Ops.H"
```

Include dependency graph for GaborPyramid.C:



## Functions

- `Image< float > optConvolve (const Image< float > &src, const Image< float > &f)`

*This is a somewhat-optimized 2-D convolution.*

- `Image< float > gabor2 (const float major_stddev, const float minor_stddev, const float period, const float phase, const float theta, int size)`

*Produces a Gabor kernel with optionally unequal major+minor axis lengths.*

- `template<class T> ImageSet< T > buildPyrGabor (const ImageSet< T > &gaussianPyr, double angle, double filter_period, double elongation, int size)`
- `template<class T> ImageSet< T > buildPyrGabor (const Image< T > &img, int depth, double angle, double filter_period, double elongation, int size)`
- `template ImageSet< float > buildPyrGabor (const Image< float > &, int, double, double, double)`



- template [ImageSet](#)< float > **buildPyrGabor** (const [ImageSet](#)< float > &, double, double, double)

### 11.133.1 Detailed Description

Definition in file [GaborPyramid.C](#).

### 11.133.2 Function Documentation

**11.133.2.1** [Image](#)<float> **gabor2** (const float *major\_stddev*, const float *minor\_stddev*, const float *period*, const float *phase*, const float *theta*, int *size*)

Produces a Gabor kernel with optionally unequal major+minor axis lengths.

Definition at line 167 of file [GaborPyramid.C](#).

References [Image](#)< T >::beginw(), [exp](#)(), [NO\\_INIT](#), [psi](#)(), and [sqrt](#)().

**11.133.2.2** [Image](#)<float> **optConvolve** (const [Image](#)< float > & *src*, const [Image](#)< float > & *f*)

This is a somewhat-optimized 2-D convolution.

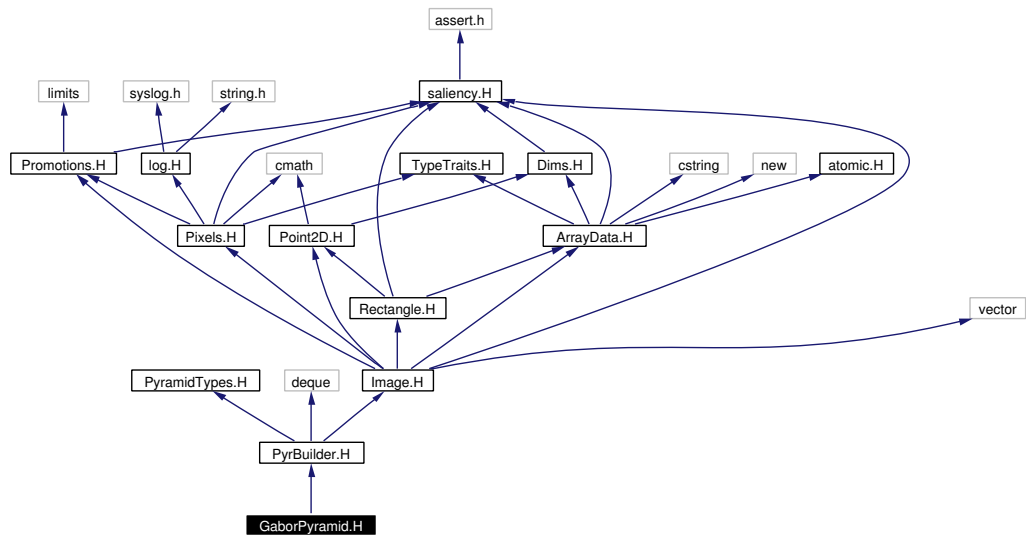
It is optimized in the sense that there are special-case inner loops for (1) cases where the filter doesn't completely overlap with the local image patch because we're at the image boundary (in this case we need a specialized inner loop to handle these boundary conditions), and (2) cases where the filter and local image patch overlap completely (in which case we can use a more efficient inner loop).

Definition at line 58 of file [GaborPyramid.C](#).

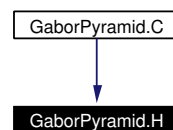
## 11.134 GaborPyramid.H File Reference

```
#include "PyrBuilder.H"
```

Include dependency graph for GaborPyramid.H:



This graph shows which files directly or indirectly include this file:



### Classes

- class [GaborPyrBuilder](#)  
*A pyramid based on convolution with oriented Gabor filters.*

### Functions

- [Image](#)< float > [gabor2](#) (const float major\_stddev, const float minor\_stddev, const float period, const float phase, const float theta, int size=-1)

*Produces a Gabor kernel with optionally unequal major+minor axis lengths.*

- `Image< float > optConvolve (const Image< float > &src, const Image< float > &f)`

*This is a somewhat-optimized 2-D convolution.*

- `template<class T> ImageSet< T > buildPyrGabor (const Image< T > &img, int depth, double angle, double filter_period, double elongation=1.0, int size=-1)`
- `template<class T> ImageSet< T > buildPyrGabor (const ImageSet< T > &gaussianPyr, double angle, double filter_period, double elongation=1.0, int size=-1)`

### 11.134.1 Detailed Description

A Gabor Pyramid

Definition in file [GaborPyramid.H](#).

### 11.134.2 Function Documentation

- 11.134.2.1 `Image<float> gabor2 (const float major_stddev, const float minor_stddev, const float period, const float phase, const float theta, int size = -1)`

Produces a Gabor kernel with optionally unequal major+minor axis lengths.

Definition at line 167 of file [GaborPyramid.C](#).

References `Image< T >::beginw()`, `exp()`, `NO_INIT`, `psi()`, and `sqrt()`.

- 11.134.2.2 `Image<float> optConvolve (const Image< float > & src, const Image< float > & f)`

This is a somewhat-optimized 2-D convolution.

It is optimized in the sense that there are special-case inner loops for (1) cases where the filter doesn't completely overlap with the local image patch because we're at the image boundary (in this case we need a specialized inner loop to handle these boundary conditions), and (2) cases where the filter and local image patch overlap completely (in which case we can use a more efficient inner loop).

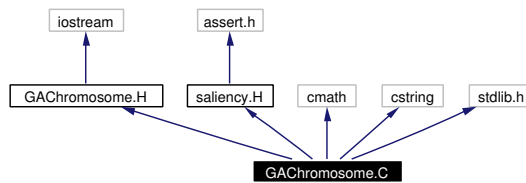
Definition at line 58 of file [GaborPyramid.C](#).

References `ASSERT`, `Image< T >::begin()`, `Image< T >::beginw()`, `Image< T >::getHeight()`, `Image< T >::getWidth()`, `Image< T >::initialized()`, and `NO_INIT`.

## 11.135 GACHromosome.C File Reference

```
#include "GACHromosome.H"
#include "saliency.H"
#include <cmath>
#include <cstring>
#include <stdlib.h>
```

Include dependency graph for GACHromosome.C:



### Functions

- `std::istream & operator>> (std::istream &in, GACHromosome &c)`  
*<< overloading.*
- `std::ostream & operator<< (std::ostream &out, GACHromosome &c)`  
*>> overloading.*

### 11.135.1 Detailed Description

A chromosome class for genetic algorithm.

Definition in file [GACHromosome.C](#).

### 11.135.2 Function Documentation

#### 11.135.2.1 `std::ostream& operator<< (std::ostream & out, GACHromosome &c)`

*>> overloading.*

Definition at line 191 of file GACHromosome.C.

### 11.135.2.2 `std::istream& operator>> (std::istream & in, GChromosome & c)`

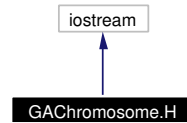
<< overloading.

Definition at line 172 of file GChromosome.C.

## 11.136 GACHromosome.H File Reference

```
#include <iostream>
```

Include dependency graph for GACHromosome.H:



This graph shows which files directly or indirectly include this file:

### Classes

- class [GACHromosome](#)  
*Chromosome class for genetic algorithm.*

### Functions

- `std::istream & operator>> (std::istream &in, GACHromosome &c)`  
*<< overloading.*
- `std::ostream & operator<< (std::ostream &out, GACHromosome &c)`  
*>> overloading.*

#### 11.136.1 Detailed Description

A chromosome class for genetic algorithm.

Definition in file [GACHromosome.H](#).

#### 11.136.2 Function Documentation

**11.136.2.1** `std::ostream& operator<< (std::ostream & out, GACHromosome &c)`

*>> overloading.*

Definition at line 191 of file GChromosome.C.

References GChromosome::breedings, GChromosome::fitness, GChromosome::genes, GChromosome::linear\_fitness, and GChromosome::size.

#### 11.136.2.2 std::istream& operator>> (std::istream & in, GChromosome & c)

<< overloading.

Definition at line 172 of file GChromosome.C.

References abs(), ASSERT, GChromosome::breedings, GChromosome::fitness, GChromosome::genes, GChromosome::linear\_fitness, GChromosome::resize(), and GChromosome::size.

## 11.137 GAMPVision.C File Reference

```
#include <iostream>
#include <fstream>
#include <cmath>
#include <mpi.h>
#include <unistd.h>
#include "saliency.H"
#include "ModelManager.H"
#include "Channels.H"
#include "VisualCortex.H"
#include "FrameSeries.H"
#include "ModelOptionDefs.H"
#include "Image_Transforms.H"
#include "Image_MathOps.H"
#include "log.H"
#include "GAPopulation.H"
```

Include dependency graph for GAMPVision.C:

### Defines

- #define **ITMAX** 400
- #define **MASTER** 0
- #define **MSG\_DATA** 0
- #define **MSG\_RESULT** 1
- #define **MSG\_STOP** 2
- #define **PATH** "/home/hpc-26/itti/openvision"
- #define **NB\_PICS** 10
- #define **NB\_FILTERS** 3
- #define **NCOEFFS** 3
- #define **POP\_SIZE** 196
- #define **RAD** 64

### Functions

- int **main** (const int argc, const char \*\*argv)



### 11.137.1 Detailed Description

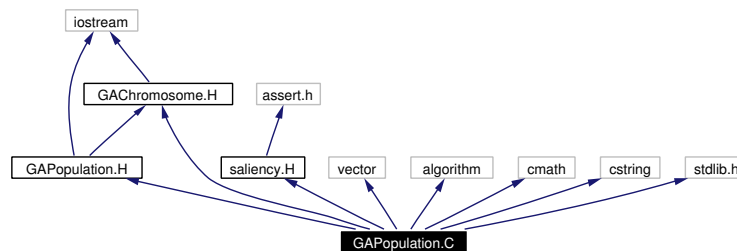
Learning filters with genetic algorithm

Definition in file [GAMPVision.C](#).

## 11.138 GAPopulation.C File Reference

```
#include "GAPopulation.H"
#include "GAChromosome.H"
#include "saliency.H"
#include <vector>
#include <algorithm>
#include <cmath>
#include <cstring>
#include <stdlib.h>
```

Include dependency graph for GAPopulation.C:



### Defines

- #define **MSG\_DATA** 1
- #define **MSG\_RESULT** 2

### Functions

- std::istream & **operator**>> (std::istream &in, [GAPopulation](#) &pop)
- std::ostream & **operator**<< (std::ostream &out, [GAPopulation](#) &pop)

### 11.138.1 Detailed Description

A population class for genetic algorithm.

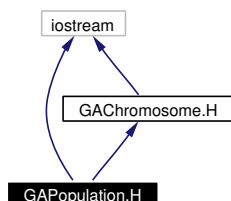
Definition in file [GAPopulation.C](#).

## 11.139 GAPopulation.H File Reference

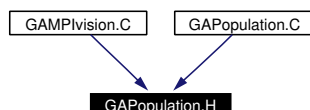
```
#include "GAChromosome.H"
```

```
#include <iostream>
```

Include dependency graph for GAPopulation.H:



This graph shows which files directly or indirectly include this file:



### Classes

- class [GAPopulation](#)  
*Population class for genetic algorithm.*

### Functions

- std::istream & [operator<<](#) (std::istream &in, [GAPopulation](#) &pop)  
*<< overloading.*
- std::ostream & [operator>>](#) (std::ostream &out, [GAPopulation](#) &pop)  
*>> overloading.*

#### 11.139.1 Detailed Description

A population class for genetic algorithm.

Definition in file [GAPopulation.H](#).

## 11.139.2 Function Documentation

### 11.139.2.1 `std::istream& operator<< (std::istream & in, GAPopulation & pop)`

`<<` overloading.

### 11.139.2.2 `std::ostream& operator>> (std::ostream & out, GAPopulation & pop)`

`>>` overloading.

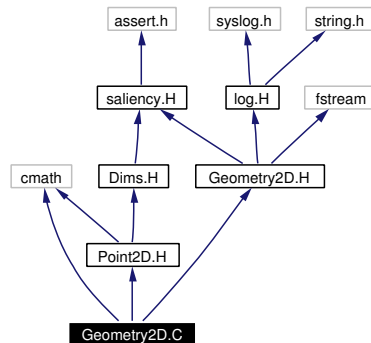
## 11.140 Geometry2D.C File Reference

```
#include "Geometry2D.H"
```

```
#include "Point2D.H"
```

```
#include <cmath>
```

Include dependency graph for Geometry2D.C:



### Functions

- bool `operator==` (const [Vector2D](#) &v1, const [Vector2D](#) &v2)  
*== operator*
- bool `operator!=` (const [Vector2D](#) &v1, const [Vector2D](#) &v2)  
*!= operator*

#### 11.140.1 Detailed Description

- classes for geometry in the plane

Definition in file [Geometry2D.C](#).

#### 11.140.2 Function Documentation

##### 11.140.2.1 bool `operator!=` (const [Vector2D](#) &v1, const [Vector2D](#) &v2)

*!= operator*

Definition at line 290 of file [Geometry2D.C](#).

**11.140.2.2 bool operator== (const [Vector2D](#) & v1, const [Vector2D](#) & v2)**

== operator

Definition at line 287 of file Geometry2D.C.

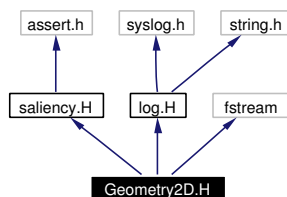
## 11.141 Geometry2D.H File Reference

```
#include "saliency.H"
```

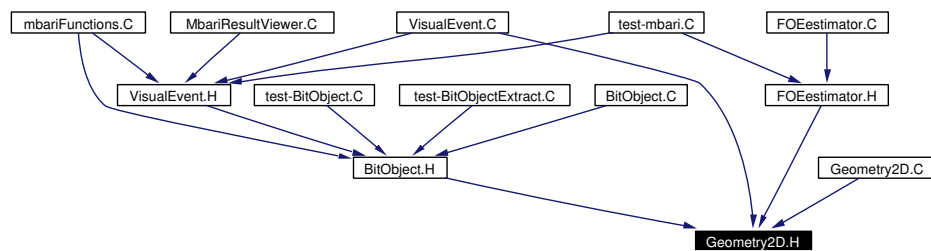
```
#include "log.H"
```

```
#include <fstream>
```

Include dependency graph for Geometry2D.H:



This graph shows which files directly or indirectly include this file:



### Classes

- class [Vector2D](#)  
*A vector in the 2D plane.*
- class [StraightLine2D](#)  
*A straight line in the 2D plane.*

### Functions

- bool [operator==](#) (const [Vector2D](#) &v1, const [Vector2D](#) &v2)  
*== operator*

- `bool operator!= (const Vector2D &v1, const Vector2D &v2)`  
*!= operator*

### 11.141.1 Detailed Description

- classes for geometry in the plane

Definition in file [Geometry2D.H](#).

### 11.141.2 Function Documentation

#### 11.141.2.1 `bool operator!= (const Vector2D & v1, const Vector2D & v2)`

*!= operator*

Definition at line 290 of file [Geometry2D.C](#).

References [Vector2D::x\(\)](#), and [Vector2D::y\(\)](#).

#### 11.141.2.2 `bool operator== (const Vector2D & v1, const Vector2D & v2)`

*== operator*

Definition at line 287 of file [Geometry2D.C](#).

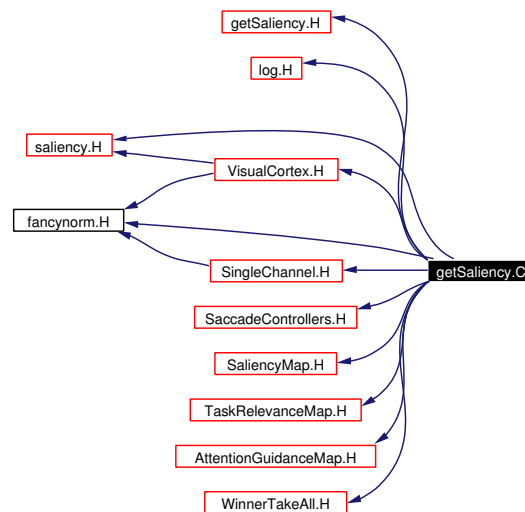
References [Vector2D::x\(\)](#), and [Vector2D::y\(\)](#).



## 11.142 getSaliency.C File Reference

```
#include "getSaliency.H"
#include "log.H"
#include "saliency.H"
#include "SaccadeControllers.H"
#include "VisualCortex.H"
#include "SingleChannel.H"
#include "fancynorm.H"
#include "SaliencyMap.H"
#include "TaskRelevanceMap.H"
#include "AttentionGuidanceMap.H"
#include "WinnerTakeAll.H"
```

Include dependency graph for getSaliency.C:



### 11.142.1 Detailed Description

Interface for obtaining lists of salient locations

////////////////////////////////////// The iLab Neuromorphic Vision C++  
Toolkit - Copyright (C) 2001 by the // University of Southern California (USC)  
and the iLab at USC. // See <http://iLab.usc.edu> for information about this

project. //////////////////////////////////////////////////////////////////// // Major portions of the  
iLab Neuromorphic Vision Toolkit are protected // under the U.S. patent “Com-  
putation of Intrinsic Perceptual Saliency // in Visual Environments, and Applica-  
tions” by Christof Koch and // Laurent Itti, California Institute of Technology,  
2001 (patent // pending; application number 09/912,225 filed July 23, 2001; see //  
<http://pair.uspto.gov/cgi-bin/final/home.pl> for current status). //  
////////////////////////////////////////////////////////////////// // This file is part of the iLab Neuro-  
morphic Vision C++ Toolkit. // // The iLab Neuromorphic Vision C++ Toolkit is free  
software; you can // redistribute it and/or modify it under the terms of the GNU General  
// Public License as published by the Free Software Foundation; either // version 2 of  
the License, or (at your option) any later version. // // The iLab Neuromorphic Vision  
C++ Toolkit is distributed in the hope // that it will be useful, but WITHOUT ANY  
WARRANTY; without even the // implied warranty of MERCHANTABILITY or FIT-  
NESS FOR A PARTICULAR // PURPOSE. See the GNU General Public License for  
more details. // // You should have received a copy of the GNU General Public License  
// along with the iLab Neuromorphic Vision C++ Toolkit; if not, write // to the Free  
Software Foundation, Inc., 59 Temple Place, Suite 330, // Boston, MA 02111-1307  
USA. //////////////////////////////////////////////////////////////////// //

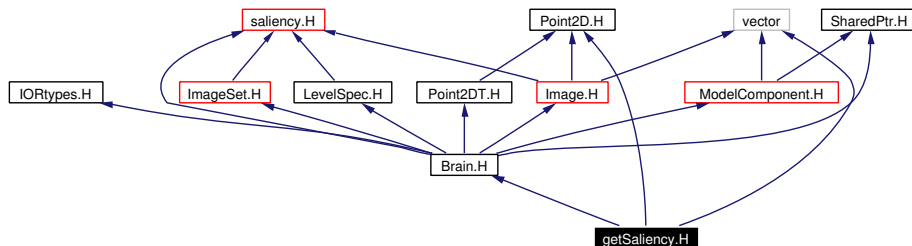
Primary maintainer for this file: Dirk Walther <[walther@caltech.edu](mailto:walther@caltech.edu)>

Definition in file [getSaliency.C](#).

## 11.143 getSaliency.H File Reference

```
#include "Brain.H"
#include "Point2D.H"
#include <vector>
```

Include dependency graph for getSaliency.H:



This graph shows which files directly or indirectly include this file:



### Classes

- class [GetSaliency](#)

*This class allows for easy access to the list of salient points in an image.*

### 11.143.1 Detailed Description

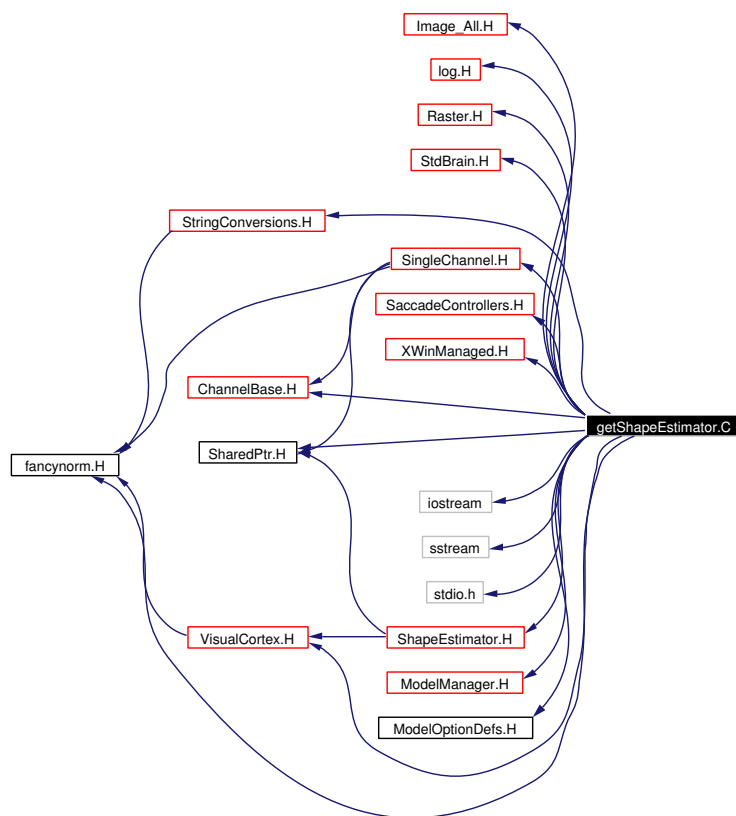
Interface for obtaining lists of salient locations

Definition in file [getSaliency.H](#).

## 11.144 getShapeEstimator.C File Reference

```
#include "Image_All.H"
#include "log.H"
#include "Raster.H"
#include "StdBrain.H"
#include "VisualCortex.H"
#include "SaccadeControllers.H"
#include "XWinManaged.H"
#include "ChannelBase.H"
#include "SharedPtr.H"
#include "fancynorm.H"
#include <iostream>
#include <sstream>
#include <stdio.h>
#include "SingleChannel.H"
#include "ModelManager.H"
#include "ModelOptionDefs.H"
#include "ShapeEstimator.H"
#include "StringConversions.H"
```

Include dependency graph for getShapeEstimator.C:



## Defines

- #define `MAX_TIME_DEFAULT` 0.7

*max time in seconds*

## Functions

- int `main` (int argc, char \*\*argv)

*The main routine.*

### 11.144.1 Detailed Description

extract size information from the SM

Definition in file [getShapeEstimator.C](#).

### 11.144.2 Define Documentation

#### 11.144.2.1 `#define MAX_TIME_DEFAULT 0.7`

max time in seconds

Definition at line 63 of file [getShapeEstimator.C](#).

### 11.144.3 Function Documentation

#### 11.144.3.1 `int main (int argc, char ** argv)`

The main routine.

Definition at line 70 of file [getShapeEstimator.C](#).

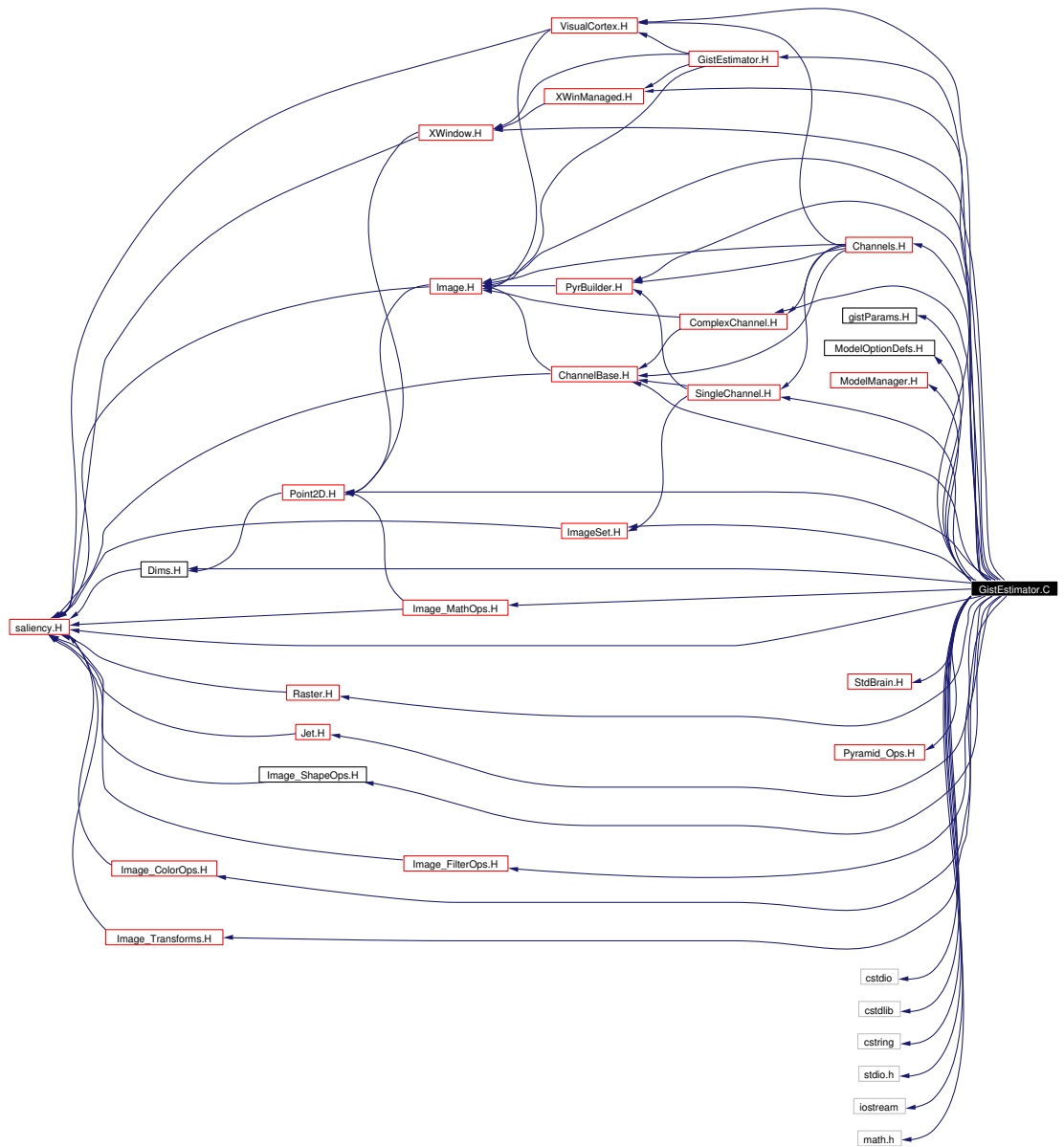
References [CloseButtonListener::add\(\)](#), [contrastModulate\(\)](#), [Image< T >::drawCircle\(\)](#), [FULLTRACE](#), [Image< T >::getDims\(\)](#), [Point2D::i](#), [Image< T >::initialized\(\)](#), [main\(\)](#), [MAX\\_TIME\\_DEFAULT](#), [MYLOGVERB](#), [OPTEXP\\_ALL](#), [Raster::ReadRGB\(\)](#), [Image< T >::resize\(\)](#), [scaleBlock\(\)](#), [CloseButtonListener::wait-ForClose\(\)](#), and [ZEROS](#).

## 11.145 GistEstimator.C File Reference

```
#include "GistEstimator.H"
#include "gistParams.H"
#include "ModelOptionDefs.H"
#include "ModelManager.H"
#include "Raster.H"
#include "XWindow.H"
#include "XWinManaged.H"
#include "saliency.H"
#include "ChannelBase.H"
#include "SingleChannel.H"
#include "ComplexChannel.H"
#include "Channels.H"
#include "VisualCortex.H"
#include "StdBrain.H"
#include "Jet.H"
#include "Image.H"
#include "Pyramid_Ops.H"
#include "Image_MathOps.H"
#include "Image_ShapeOps.H"
#include "Image_FilterOps.H"
#include "Image_ColorOps.H"
#include "Image_Transforms.H"
#include "ImageSet.H"
#include "PyrBuilder.H"
#include "Point2D.H"
#include "Dims.H"
#include <stdio>
#include <stdlib>
#include <string>
```

```
#include <stdio.h>
#include <iostream>
#include <math.h>
```

Include dependency graph for GistEstimator.C:





### 11.145.1 Detailed Description

extract estimated gist using available features of the image

Definition in file [GistEstimator.C](#).

## 11.146 GistEstimator.H File Reference

```
#include "ModelComponent.H"  
#include "SharedPtr.H"  
#include "Image.H"  
#include "VisualCortex.H"  
#include "XWindow.H"  
#include "XWinManaged.H"
```

Include dependency graph for GistEstimator.H:

This graph shows which files directly or indirectly include this file:

### Classes

- class [GistEstimator](#)

### 11.146.1 Detailed Description

extract estimated gist using available features of the image

Definition in file [GistEstimator.H](#).

## 11.147 GistEstimatorConfigurator.C File Reference

```
#include "GistEstimatorConfigurator.H"
```

```
#include "GistEstimator.H"
```

```
#include "ModelManager.H"
```

```
#include "ModelOptionDefs.H"
```

Include dependency graph for GistEstimatorConfigurator.C:

### 11.147.1 Detailed Description

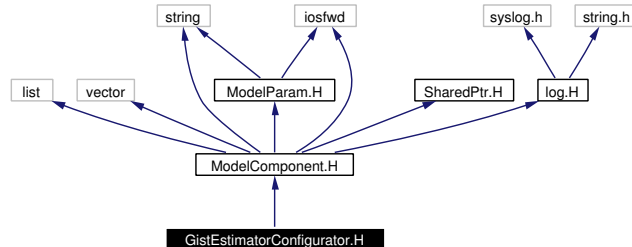
future expansions for run-time options

Definition in file [GistEstimatorConfigurator.C](#).

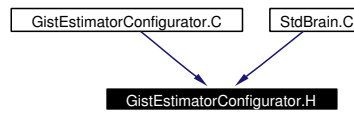
## 11.148 GistEstimatorConfigurator.H File Reference

```
#include "ModelComponent.H"
```

Include dependency graph for GistEstimatorConfigurator.H:



This graph shows which files directly or indirectly include this file:



### Classes

- class [GistEstimatorConfigurator](#)  
*GistEstimator* configurator.

### 11.148.1 Detailed Description

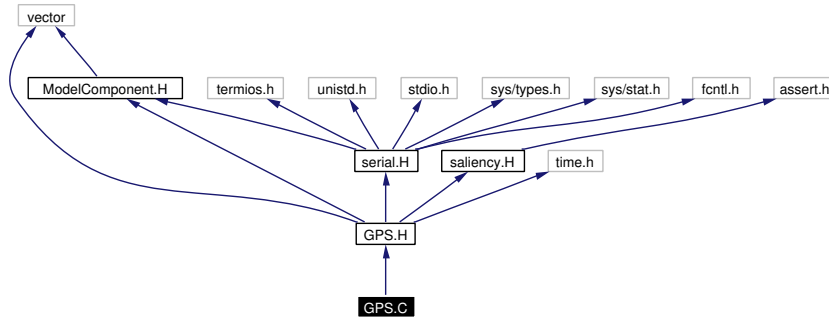
future expansions for run-time options

Definition in file [GistEstimatorConfigurator.H](#).

## 11.149 GPS.C File Reference

```
#include "GPS.H"
```

Include dependency graph for GPS.C:



### Variables

- **byte** `wayPoint1` [ ] = "\x0A\x02\x07\x00"
- **byte** `wayPoint2` [ ] = "\x06\x02\x23\x00"
- **byte** `test` [ ] = "\x1c\x02\x00\x00"
- **byte** `tim1` [ ] = "\x0a\x02\x05\x00"
- **byte** `loc1` [ ] = "\x0a\x02\x02\x00"
- **byte** `pvtON` [ ] = "\x0a\x02\x31\x00"
- **byte** `pvtOFF` [ ] = "\x0a\x02\x32\x00"

### 11.149.1 Detailed Description

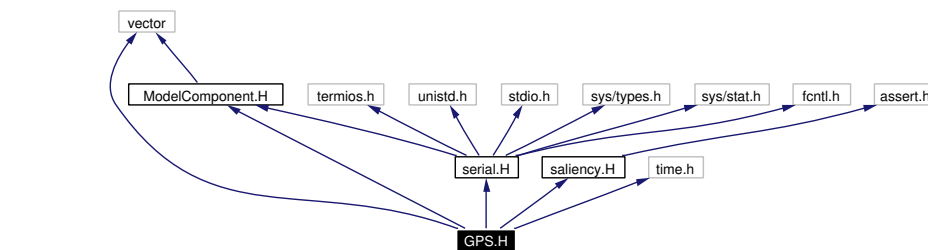
[Serial](#) interface to a GPS unit

Definition in file [GPS.C](#).

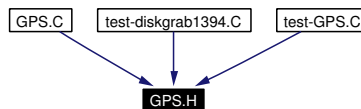
## 11.150 GPS.H File Reference

```
#include "serial.H"
#include "saliency.H"
#include "ModelComponent.H"
#include <time.h>
#include <vector>
```

Include dependency graph for GPS.H:



This graph shows which files directly or indirectly include this file:



## Classes

- struct **WayPoint**
- struct **GPSTLocation**
- struct **GPSPvtDataType**
- class [GpsDev](#)

*Interface with a GPS unit.*

## Typedefs

- typedef GPSPvtDataType **GPSPvtData**

## Variables

- GPSPvtDataType **packed**

### 11.150.1 Detailed Description

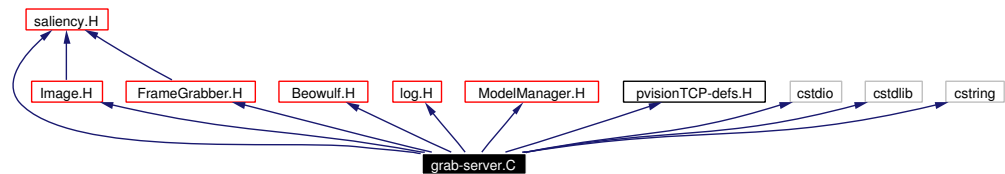
[Serial](#) interface to a GPS unit

Definition in file [GPS.H](#).

## 11.151 grab-server.C File Reference

```
#include "saliency.H"  
#include "Image.H"  
#include "FrameGrabber.H"  
#include "Beowulf.H"  
#include "log.H"  
#include "ModelManager.H"  
#include "pvisionTCP-defs.H"  
#include <stdio>  
#include <stdlib>  
#include <string>
```

Include dependency graph for grab-server.C:



## Functions

- `int main (const int argc, const char **argv)`

### 11.151.1 Detailed Description

A video frame grab server for [Beowulf](#)

Definition in file [grab-server.C](#).

### 11.151.2 Function Documentation

#### 11.151.2.1 `int main (const int argc, const char ** argv)`

This simple executable grabs frames and makes them available for clients connected to it through our [Beowulf](#) class.



Definition at line 51 of file grab-server.C.

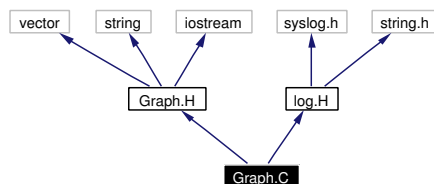
References TCPmessage::addImage(), BEO\_RETINA, SharedPtr< T >::get(), main(), and TCPmessage::reset().

## 11.152 Graph.C File Reference

```
#include "Graph.H"
```

```
#include "log.H"
```

Include dependency graph for Graph.C:



### 11.152.1 Detailed Description

This has methods to print entity

Definition in file [Graph.C](#).

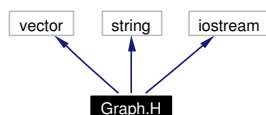
## 11.153 Graph.H File Reference

```
#include <vector>
```

```
#include <string>
```

```
#include <iostream>
```

Include dependency graph for Graph.H:



This graph shows which files directly or indirectly include this file:

### Classes

- class [Property](#)  
*Definition of Property class and its functions.*
- class [Entity](#)  
*Definition of Entity class and its functions.*
- class [Entity\\_TWM](#)  
*Definition of Entity\_TWM class and its functions.*
- class [Edge](#)  
*Definition of Edge class and its functions.*

### Defines

- #define **SIZE** 6
- #define **PROP\_SIZE** 3
- #define **GRANULARITY0** 0.5
- #define **GRANULARITY1** 1
- #define **GRANULARITY2** 0.5
- #define **GRANULARITY3** 1
- #define **GRANULARITY4** 1
- #define **GRANULARITY5** 1

### 11.153.1 Detailed Description

This has definitions of nodes of ontology and task graph

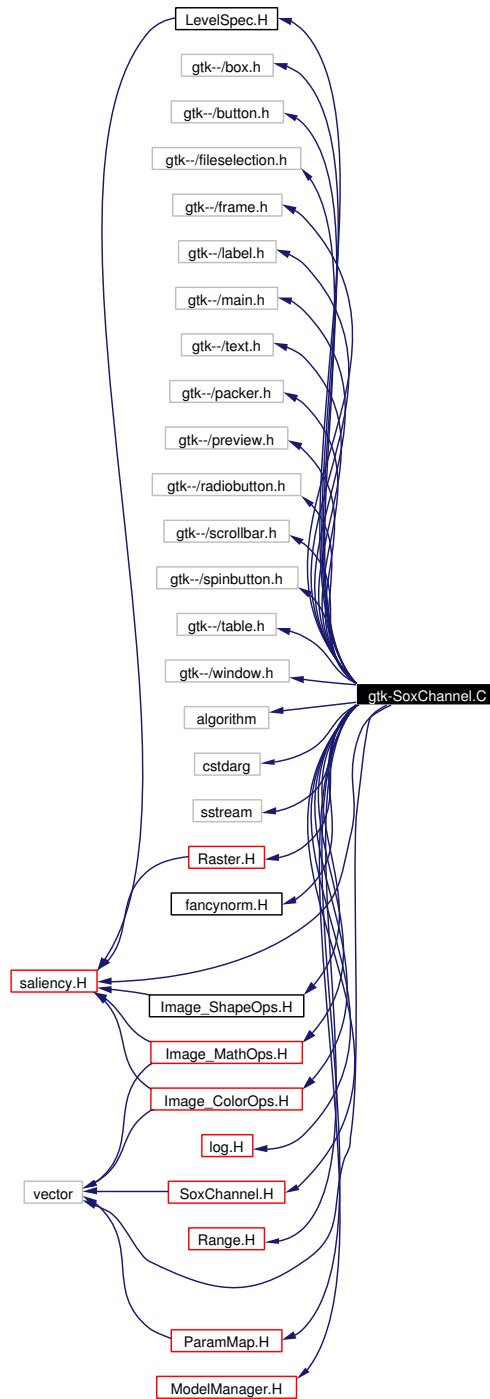
Definition in file [Graph.H](#).

## 11.154 gtk-SoxChannel.C File Reference

```
#include "saliency.H"
#include <gtk-/box.h>
#include <gtk-/button.h>
#include <gtk-/fileselection.h>
#include <gtk-/frame.h>
#include <gtk-/label.h>
#include <gtk-/main.h>
#include <gtk-/text.h>
#include <gtk-/packer.h>
#include <gtk-/preview.h>
#include <gtk-/radiobutton.h>
#include <gtk-/scrollbar.h>
#include <gtk-/spinbutton.h>
#include <gtk-/table.h>
#include <gtk-/window.h>
#include <algorithm>
#include <cstdarg>
#include <sstream>
#include <vector>
#include "fancynorm.H"
#include "Image_ColorOps.H"
#include "Image_MathOps.H"
#include "Image_ShapeOps.H"
#include "LevelSpec.H"
#include "log.H"
#include "ParamMap.H"
#include "Range.H"
#include "Raster.H"
#include "SoxChannel.H"
```

```
#include "ModelManager.H"
```

Include dependency graph for gtk-SoxChannel.C:



## Classes

- class **ExtTable**
- class **SoxWindow**

## Functions

- void **copyToPreview** (const [Image](#)< [PixRGB](#)< [byte](#) > > &img, Gtk::Preview &pv)
- int **main** (int argc, char \*\*argv)

### 11.154.1 Detailed Description

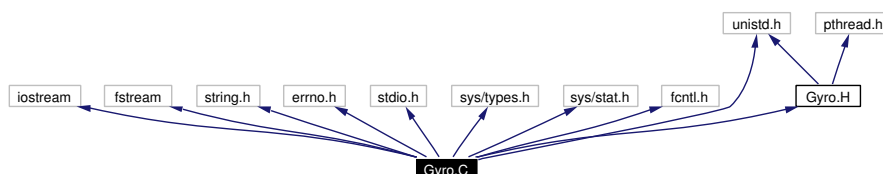
This is a Gtk+ (written using the Gtk- C++ API) gui interface to the [SoxChannel](#) class.  
Definition in file [gtk-SoxChannel.C](#).



## 11.155 Gyro.C File Reference

```
#include <iostream>
#include <fstream>
#include <string.h>
#include <errno.h>
#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
#include "Gyro.H"
```

Include dependency graph for Gyro.C:



### Defines

- `#define XANGLE_CONVERSION .0432`
- `#define YANGLE_CONVERSION .0405`

### Functions

- `void * Gyro_run (void *c0)`

#### 11.155.1 Detailed Description

calculates angle of mouse with respect to floor

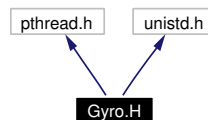
Definition in file [Gyro.C](#).

## 11.156 Gyro.H File Reference

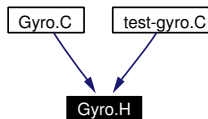
```
#include <pthread.h>
```

```
#include <unistd.h>
```

Include dependency graph for Gyro.H:



This graph shows which files directly or indirectly include this file:



### Classes

- class **Gyro**

#### 11.156.1 Detailed Description

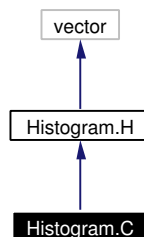
Interface to a GyroMouse

Definition in file [Gyro.H](#).

## 11.157 Histogram.C File Reference

```
#include "Histogram.H"
```

Include dependency graph for Histogram.C:



### 11.157.1 Detailed Description

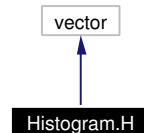
[Histogram](#)

Definition in file [Histogram.C](#).

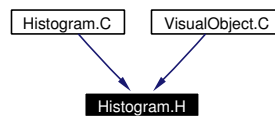
## 11.158 Histogram.H File Reference

```
#include <vector>
```

Include dependency graph for Histogram.H:



This graph shows which files directly or indirectly include this file:



### Classes

- class [Histogram](#)  
*Histogram.*

### 11.158.1 Detailed Description

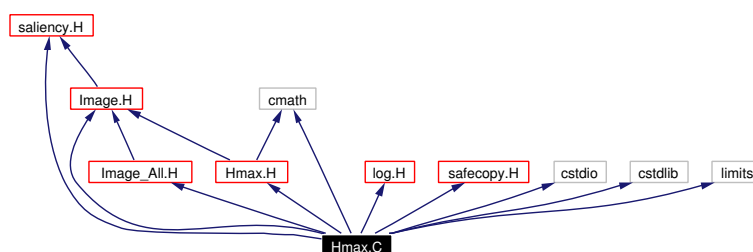
Header file for histograms

Definition in file [Histogram.H](#).

## 11.159 Hmax.C File Reference

```
#include "Hmax.H"
#include "Image.H"
#include "Image_All.H"
#include "log.H"
#include "safecopy.H"
#include "saliency.H"
#include <cmath>
#include <cstdio>
#include <cstdlib>
#include <limits>
```

Include dependency graph for Hmax.C:



### 11.159.1 Detailed Description

Riesenhuber &amp; Poggio's HMAX model for object recognition

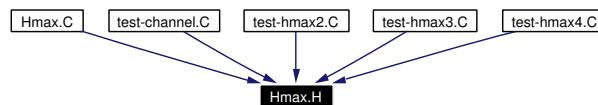
Definition in file [Hmax.C](#).

## 11.160 Hmax.H File Reference

```
#include "Image.H"  
#include <vector>  
#include <cmath>  
#include <iostream>  
#include <fstream>  
#include <string>  
#include <dirent.h>
```

Include dependency graph for Hmax.H:

This graph shows which files directly or indirectly include this file:



### Classes

- class [Hmax](#)

*Object recognition model inspired from Riesenhuber & Poggio's HMAX model.*

### 11.160.1 Detailed Description

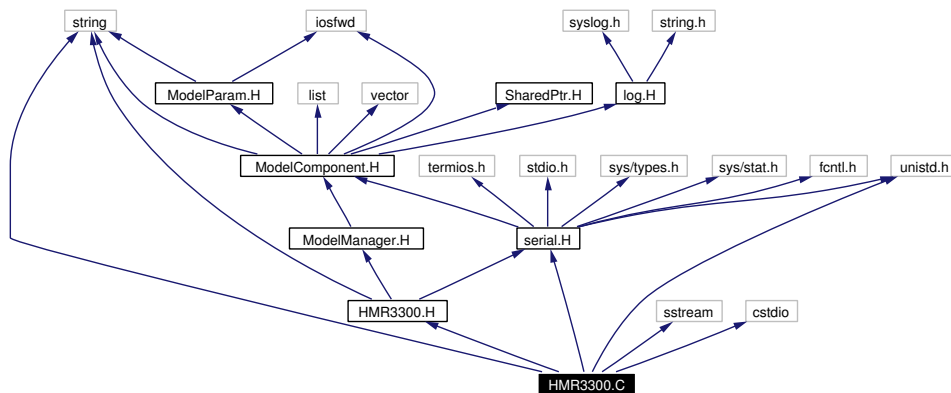
Riesenhuber & Poggio's HMAX model for object recognition

Definition in file [Hmax.H](#).

## 11.161 HMR3300.C File Reference

```
#include "HMR3300.H"
#include "serial.H"
#include <sstream>
#include <string>
#include <cstdio>
#include <unistd.h>
```

Include dependency graph for HMR3300.C:



### Functions

- void \* **Compass\_run** (void \*c)

#### 11.161.1 Detailed Description

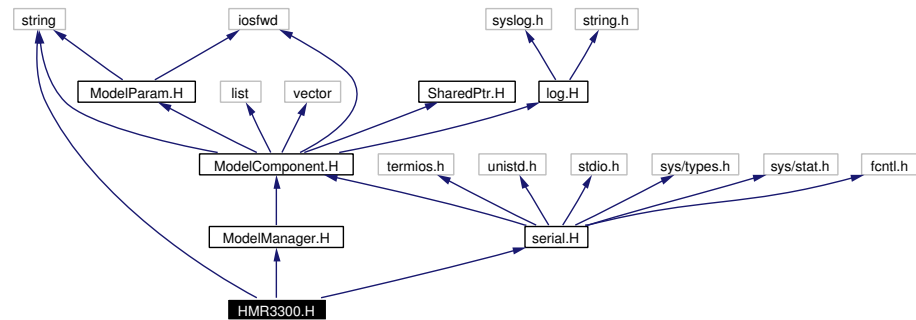
class for interfacing with a Honeywell 3300 compass

Definition in file [HMR3300.C](#).

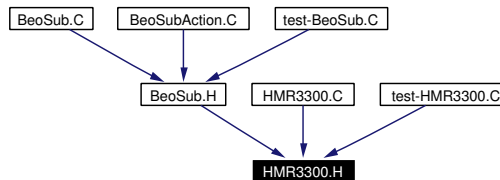
## 11.162 HMR3300.H File Reference

```
#include <string>
#include "ModelManager.H"
#include "serial.H"
```

Include dependency graph for HMR3300.H:



This graph shows which files directly or indirectly include this file:



### Classes

- class [Heading](#)  
*A simple class to represent 3D heading information.*
- class [HMR3300](#)  
*Class for interfacing with a Honeywell 3300 compass.*

### 11.162.1 Detailed Description

class for interfacing with a Honeywell 3300 compass

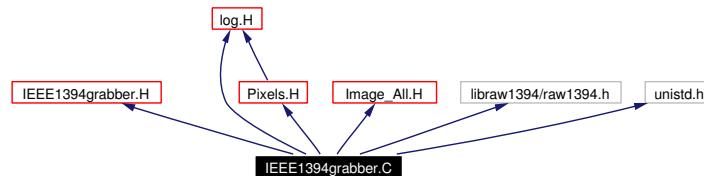


Definition in file [HMR3300.H](#).

## 11.163 IEEE1394grabber.C File Reference

```
#include "IEEE1394grabber.H"  
#include "log.H"  
#include "Pixels.H"  
#include "Image_All.H"  
#include <libraw1394/raw1394.h>  
#include <unistd.h>
```

Include dependency graph for IEEE1394grabber.C:



### Defines

- #define IEEE1394WAIT 1

### 11.163.1 Detailed Description

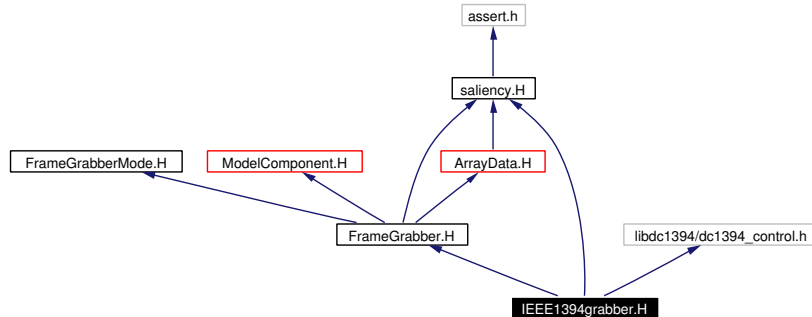
Interface with a FireWire digital camera

Definition in file [IEEE1394grabber.C](#).

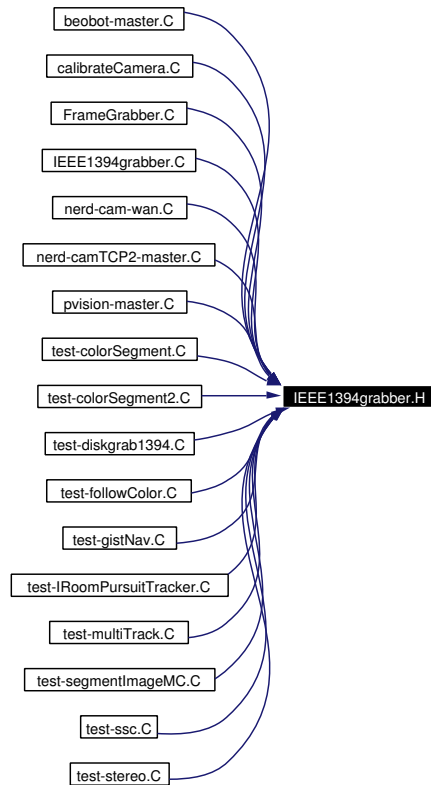
## 11.164 IEEE1394grabber.H File Reference

```
#include "saliency.H"  
#include <libdc1394/dc1394_control.h>  
#include "FrameGrabber.H"
```

Include dependency graph for IEEE1394grabber.H:



This graph shows which files directly or indirectly include this file:



## Classes

- class [IEEE1394grabber](#)

*Definition and access functions for FireWire digital camera frame capture.*

## Defines

- #define [IEEE1394GRABSPEED](#) SPEED\_400

*Firewire bus speed to use.*

- #define [IEEE1394GRABFORMAT](#) FORMAT\_VGA\_NONCOMPRESSED

*Firewire video format.*

### 11.164.1 Detailed Description

Definition and access functions for FireWire grabber

Definition in file [IEEE1394grabber.H](#).

### 11.164.2 Define Documentation

#### 11.164.2.1 **#define IEEE1394GRABFORMAT FORMAT\_VGA\_-NONCOMPRESSED**

Firewire video format.

Definition at line 52 of file IEEE1394grabber.H.

Referenced by IEEE1394grabber::start1().

#### 11.164.2.2 **#define IEEE1394GRABSPEED SPEED\_400**

Firewire bus speed to use.

Definition at line 49 of file IEEE1394grabber.H.

Referenced by IEEE1394grabber::start1().

## 11.165 Image.C File Reference

```
#include "Image.H"
#include "Image_ShapeOps.H"
#include "log.H"
#include "safecopy.H"
#include "saliency.H"
#include "Pixels.H"
#include <algorithm>
#include <cmath>
#include <functional>
#include "font.c"
```

Include dependency graph for Image.C:

### Defines

- #define **LERP**(a, l, h) ((l)+(((h)-(l))\*a))

### 11.165.1 Detailed Description

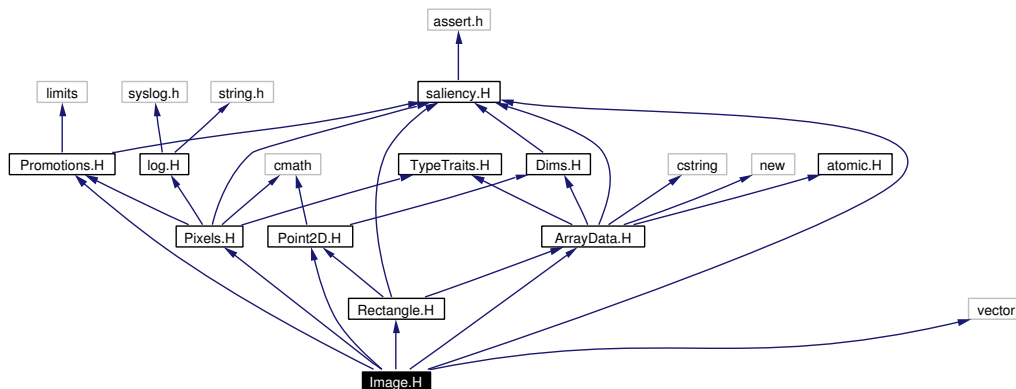
An image template class with many image processing functions

Definition in file [Image.C](#).

## 11.166 Image.H File Reference

```
#include "saliency.H"
#include "Promotions.H"
#include "ArrayData.H"
#include "Point2D.H"
#include "Rectangle.H"
#include "Pixels.H"
#include <vector>
```

Include dependency graph for Image.H:



This graph shows which files directly or indirectly include this file:

### Classes

- class [Image](#)  
*Generic image template class.*

### 11.166.1 Detailed Description

An image template class with many image processing functions

Definition in file [Image.H](#).

## 11.167 Image\_All.H File Reference

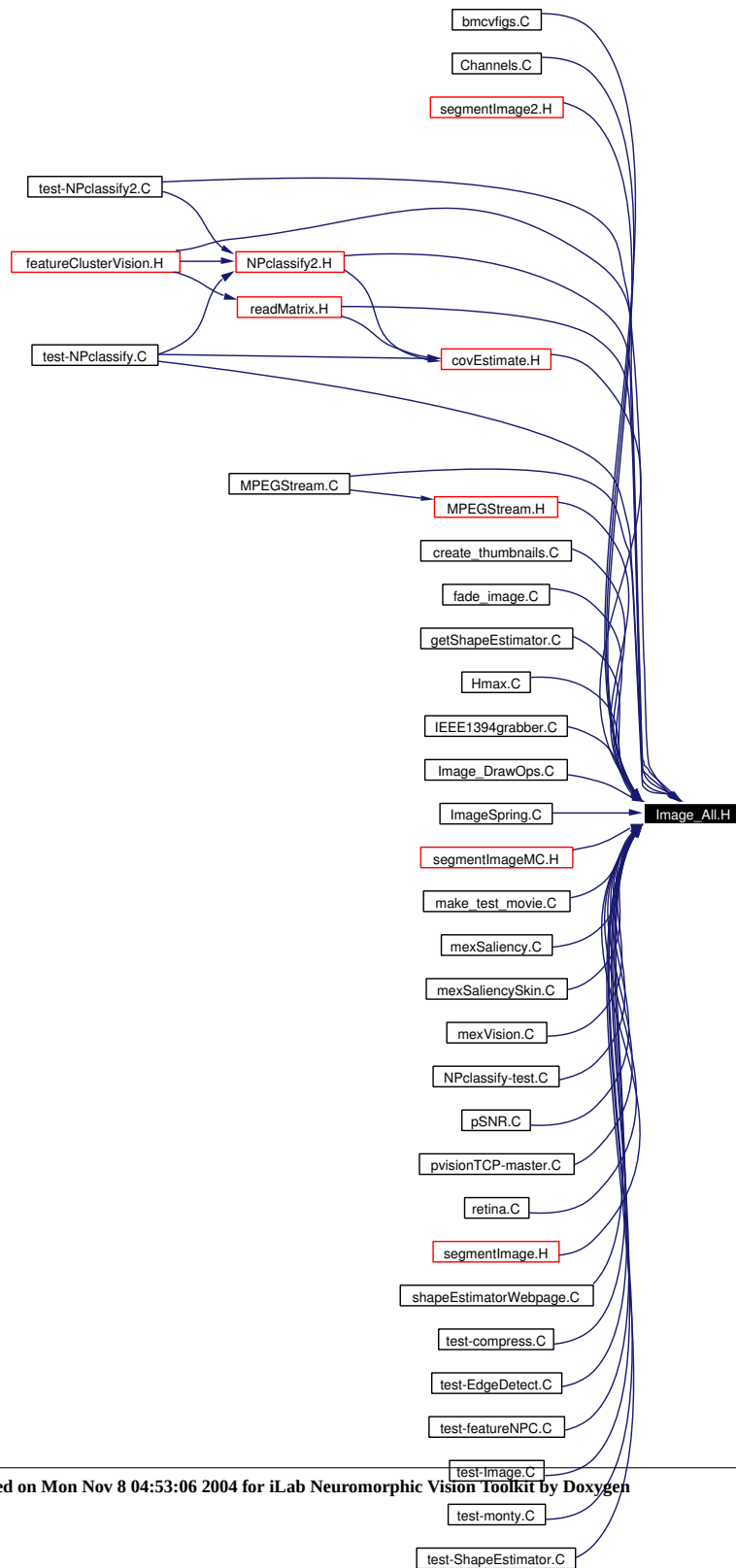
```
#include "Image.H"  
#include "Image_ColorOps.H"  
#include "Image_DrawOps.H"  
#include "Image_FilterOps.H"  
#include "Image_IO.H"  
#include "Image_MathOps.H"  
#include "Image_ShapeOps.H"  
#include "Image_Transforms.H"
```

Include dependency graph for Image\_All.H:



This graph shows which files directly or indirectly include this file:





### 11.167.1 Detailed Description

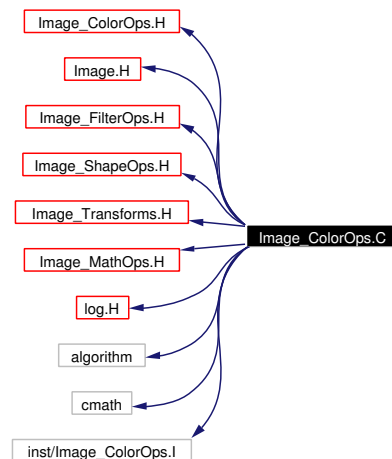
Shorthand to include all [Image](#) methods

Definition in file [Image\\_All.H](#).

## 11.168 Image\_ColorOps.C File Reference

```
#include "Image_ColorOps.H"
#include "Image.H"
#include "Image_FilterOps.H"
#include "Image_ShapeOps.H"
#include "Image_Transforms.H"
#include "Image_MathOps.H"
#include "log.H"
#include <algorithm>
#include <cmath>
#include "inst/Image_ColorOps.I"
```

Include dependency graph for Image\_ColorOps.C:



## Functions

- `template<class T> Image< PixRGB< T > > makeRGB (const Image< T > &red, const Image< T > &green, const Image< T > &blue)`  
*Set the red, green and blue components from 3 monochromatic images.*
- `template<class T> void getComponents (const Image< PixRGB< T > > &src, Image< T > &red, Image< T > &green, Image< T > &blue)`  
*Get the red, green and blue components as 3 monochromatic images.*

- `Image< PixRGB< float > > normalizeRGPolar` (const `Image< float > &src`, const float max, const float min)
- `template<class T> Image< PixRGB< T > > stain` (const `Image< T > &src`, `PixRGB< float > color`)  
*Create a color image by staining a monochrome image.*
- `Image< PixRGB< float > > overlayStain` (const `Image< float > &top`, const `Image< float > &bottom`, const float trans, const char channel)  
*Overlay an image over another, however, stain the top image.*
- `template<class T> void getMinMaxC` (const `Image< PixRGB< T > > &src`, `T &mi`, `T &ma`)  
*Get min and max of the components.*
- `template<class T> Image< T > luminance` (const `Image< PixRGB< T > > &src`)  
*Compute luminance of a color image.*
- `template<class T> Image< T > luminance` (const `Image< T > &src`)  
*Compute luminance of a greyscale image (no-op).*
- `template<class T> Image< PixRGB< T > > toRGB` (const `Image< T > &src`)  
*Convert grayscale image to RGB.*
- `template<class T> Image< PixRGB< T > > toRGB` (const `Image< PixRGB< T > > &src`)  
*Convert RGB image to RGB (no-op).*
- `template<class T> Image< PixRGB< T > > warp3D` (const `Image< PixRGB< T > > &ima`, const `Image< float > &zmap`, const float pitch, const float yaw, const float zMax, `Dims &dims`)  
*Warp image in 3D onto z-map.*
- `template<class T> Image< float > infoMeasure` (const `Image< PixRGB< T > > &src`, const float eps, const int size)  
*Compute color information measure based in infoFFT.*
- `template<class T> void getYIQ` (const `Image< PixRGB< T > > &src`, `Image< T > &y`, `Image< T > &i`, `Image< T > &q`)  
*Get YIQ color components.*

- `template<class T> void getYUV (const Image< PixRGB< T > > &src, Image< T > &y, Image< T > &u, Image< T > &v)`  
*Get YUV color components from an RGB image.*
- `template<class T> Image< PixRGB< T > > setYUV (const Image< T > &y, const Image< T > &u, const Image< T > &v)`  
*Build an RGB image from YUV color components.*
- `template<class T> Image< PixRGB< T > > fromRGB (const T *data, const Dims &dims)`  
*Build an RGB image from an RGB C array.*
- `template<class T> Image< PixRGB< T > > fromARGB (const T *data, const Dims &dims)`  
*Build an RGB image from a alpha+RGB C array.*
- `template<class T> Image< PixRGB< T > > fromBGR (const T *data, const Dims &dims)`  
*Build an RGB image from a BGR C array.*
- `template<class T> Image< PixRGB< T > > fromBGRA (const T *data, const Dims &dims)`  
*Build an RGB image from a BGR+alpha C array.*
- `Image< PixRGB< byte > > fromRGB555 (const byte *data, const Dims &dims)`  
*Build an RGB image from an RGB555 C array.*
- `Image< PixRGB< byte > > fromRGB555bs (const byte *data, const Dims &dims)`  
*Build an RGB image from an RGB555 C array, byte-swapping in the process.*
- `Image< PixRGB< byte > > fromRGB565 (const byte *data, const Dims &dims)`  
*Build an RGB image from an RGB565 C array.*
- `Image< PixRGB< byte > > fromRGB565bs (const byte *data, const Dims &dims)`  
*Build an RGB image from an RGB565 C array, byte-swapping in the process.*
- `template<class T> Image< PixRGB< T > > fromYUV444 (const T *data, const Dims &dims)`  
*Build an RGB image from a YUV444 C array.*

- `template<class T> Image< PixRGB< T > > fromYUV444bs` (const T \*data, const Dims &dims)  
*Build an RGB image from a YUV444 C array, byte-swapping in the process.*
- `template<class T> Image< PixRGB< T > > fromYUV422` (const T \*data, const Dims &dims)  
*Build an RGB image from a YUV422 C array.*
- `void toYUV422` (const Image< PixRGB< byte > > img, byte \*y, byte \*u, byte \*v, const int ystride, const int ustride, const int vstride)  
*Fill pre-allocated YUV422 C arrays with values from an image.*
- `template<class T> Image< PixRGB< T > > fromYUV422bs` (const T \*data, const Dims &dims)  
*Build an RGB image from a YUV422 C array, byte-swapping in the process.*
- `template<class T> Image< PixRGB< T > > fromYUV411` (const T \*data, const Dims &dims)  
*Build an RGB image from a YUV411 C array.*
- `template<class T> Image< PixRGB< T > > fromYUV411bs` (const T \*data, const Dims &dims)  
*Build an RGB image from a YUV411 C array, byte-swapping in the process.*
- `template<class T> Image< PixRGB< T > > fromYUV444P` (const T \*data, const Dims &dims)  
*Build an RGB image from a YUV422-planar C array.*
- `template<class T> Image< PixRGB< T > > fromYUV422P` (const T \*data, const Dims &dims)  
*Build an RGB image from a YUV422-planar C array.*
- `template<class T> Image< PixRGB< T > > fromYUV411P` (const T \*data, const Dims &dims)  
*Build an RGB image from a YUV411-planar C array.*
- `template<class T> Image< PixRGB< T > > fromYUV420P` (const T \*data, const Dims &dims)  
*Build an RGB image from a YUV420-planar C array.*
- `template<class T> Image< PixRGB< T > > fromYUV410P` (const T \*data, const Dims &dims)

*Build an RGB image from a YUV410-planar C array.*

- `template<class T> Image< PixRGB< T > > fromMono (const T *data, const Dims &dims)`

*Build an RGB image from a grayscale C array of size dims.w()\*dims.h().*

- `template<class T> Image< PixRGB< T > > luminanceNormalize (const Image< PixRGB< T > > &src, const T thresh)`

*Normalize my values by those in the lum image.*

- `template<class T> void getRGBY (const Image< PixRGB< T > > &src, Image< typename promote_trait< T, float >::TP > &rg, Image< typename promote_trait< T, float >::TP > &by, const T thresh)`

*Compute R/G and B/Y channels from a luminanceNormalized color image.*

- `template<class T> void getRGBYsimple (const Image< PixRGB< T > > &src, Image< typename promote_trait< T, float >::TP > &rg, Image< typename promote_trait< T, float >::TP > &by, const T thresh)`

*Compute R/G and B/Y channels in a simpler way.*

- `template<class T> void getRGBY (const Image< PixRGB< T > > &src, Image< T > &rr, Image< T > &gg, Image< T > &bb, Image< T > &yy, const T thresh)`

*Compute R, G, B, Y channels from a luminanceNormalized color image.*

- `Image< PixRGB< byte > > contrastModulate (const Image< PixRGB< byte > > &img, const Image< float > &mask, float baseContrast, byte baseBright)`

*contrast-modulate an RGB image with a mask*

- `template<class T> double pSNRcolor (const Image< PixRGB< T > > &img1, const Image< PixRGB< T > > &img2)`

*Compute peak signal-to-noise ratio between two color images.*

- `template<class T> double pSNRcolor (const Image< PixRGB< T > > &img1, const Image< PixRGB< T > > &img2, const Image< float > &weight)`

*Compute weighted peak signal-to-noise ratio between two color images.*

- `template<class T> Image< PixRGB< T > > normalizeRGB (const Image< PixRGB< T > > &img, PixRGB< T > min, PixRGB< T > max)`

*Normalize each component of the RGB image separately.*

- `template<class T> Image< T > maxRGB (const Image< PixRGB< T > > &img)`

*Return the pixel-wise maximum of the r, g and b component.*

- `template<class T> Image< PixRGB< T > > colorStain (const Image< T > &src, const T &min, const T &max, const PixRGB< T > &color)`

*Stain a black and white image with a color.*

- `void RGBtoCIE (const PixRGB< byte > &rgbColor, float &cr, float &cg, float &intens)`

*convert RGB color values into C.I.E. coordinates*

- `Image< float > hueDistance (const Image< PixRGB< byte > > &img, float muR, float muG, float sigR, float sigG, float rho)`

*distance between hue of image pixels and a given hue*

### 11.168.1 Detailed Description

Color operations on [Image](#)

Definition in file [Image\\_ColorOps.C](#).

### 11.168.2 Function Documentation

- 11.168.2.1** `template<class T> Image< PixRGB<T> > colorStain (const Image< T > & src, const T & min, const T & max, const PixRGB< T > & color)`

Stain a black and white image with a color.

**Parameters:**

**src** the black and white image to be stained

**min** This value will be converted to 0 (black)

**max** This value will be converted to color

**color** The color used for staining

Definition at line 1261 of file [Image\\_ColorOps.C](#).

References [ASSERT](#), [Image< T >::begin\(\)](#), [Image< T >::beginw\(\)](#), [Image< T >::end\(\)](#), [Image< T >::getDims\(\)](#), and [NO\\_INIT](#).

Referenced by [main\(\)](#).



### 11.168.2.2 **Image< PixRGB<byte> > contrastModulate (const Image< PixRGB< byte > > & img, const Image< float > & mask, float baseContrast, byte baseBright)**

contrast-modulate an RGB image with a mask

Definition at line 1183 of file Image\_ColorOps.C.

References ASSERT, Image< T >::clamp(), and Image< T >::normalize().

Referenced by SimulationViewerStd::getTraj(), and main().

### 11.168.2.3 **template<class T> Image<PixRGB<T> > fromARGB (const T \* data, const Dims & dims)**

Build an RGB image from a alpha+RGB C array.

Definition at line 528 of file Image\_ColorOps.C.

References Image< T >::beginw(), Image< T >::endw(), and NO\_INIT.

Referenced by FrameGrabber::grab2rgb().

### 11.168.2.4 **template<class T> Image<PixRGB<T> > fromBGR (const T \* data, const Dims & dims)**

Build an RGB image from a BGR C array.

Definition at line 546 of file Image\_ColorOps.C.

References Image< T >::beginw(), Image< T >::endw(), and NO\_INIT.

Referenced by FrameGrabber::grab2rgb().

### 11.168.2.5 **template<class T> Image<PixRGB<T> > fromBGRA (const T \* data, const Dims & dims)**

Build an RGB image from a BGR+alpha C array.

Definition at line 564 of file Image\_ColorOps.C.

References Image< T >::beginw(), Image< T >::endw(), and NO\_INIT.

Referenced by FrameGrabber::grab2rgb().

### 11.168.2.6 **template<class T> Image<PixRGB<T> > fromMono (const T \* data, const Dims & dims)**

Build an RGB image from a grayscale C array of size dims.w()\*dims.h().

Definition at line 975 of file Image\_ColorOps.C.

References Image< T >::beginw(), Image< T >::endw(), and NO\_INIT.

Referenced by FrameGrabber::grab2rgb().

#### 11.168.2.7 **template<class T> Image<PixRGB<T> > fromRGB (const T \* data, const Dims & dims)**

Build an RGB image from an RGB C array.

This will operate a deep copy of the data; this function is somewhat redundant with the Image constructor and is provided for completeness with respect to the fromXXX methods only. Note that data should have 3\*dims.w()\*dims.h() elements, and it is the responsibility of the caller to free that data later.

Definition at line 518 of file Image\_ColorOps.C.

References Image< T >::getArrayPtr(), Dims::h(), NO\_INIT, and Dims::w().

Referenced by FrameGrabber::grab2rgb().

#### 11.168.2.8 **Image<PixRGB<byte> > fromRGB555 (const byte \* data, const Dims & dims)**

Build an RGB image from an RGB555 C array.

Definition at line 581 of file Image\_ColorOps.C.

References Image< T >::beginw(), byte, Image< T >::endw(), and NO\_INIT.

Referenced by FrameGrabber::grab2rgb().

#### 11.168.2.9 **Image<PixRGB<byte> > fromRGB555bs (const byte \* data, const Dims & dims)**

Build an RGB image from an RGB555 C array, byte-swapping in the process.

Definition at line 602 of file Image\_ColorOps.C.

References Image< T >::beginw(), byte, Image< T >::endw(), and NO\_INIT.

Referenced by FrameGrabber::grab2rgb().

#### 11.168.2.10 **Image<PixRGB<byte> > fromRGB565 (const byte \* data, const Dims & dims)**

Build an RGB image from an RGB565 C array.

Definition at line 623 of file Image\_ColorOps.C.

References Image< T >::beginw(), byte, Image< T >::endw(), and NO\_INIT.

Referenced by FrameGrabber::grab2rgb().

#### 11.168.2.11 **Image<PixRGB<byte> > fromRGB565bs (const byte \* data, const Dims & dims)**

Build an RGB image from an RGB565 C array, byte-swapping in the process.

Definition at line 644 of file Image\_ColorOps.C.

References Image< T >::beginw(), byte, Image< T >::endw(), and NO\_INIT.

Referenced by FrameGrabber::grab2rgb().

#### 11.168.2.12 **template<class T> Image<PixRGB<T> > fromYUV410P (const T \* data, const Dims & dims)**

Build an RGB image from a YUV410-planar C array.

Definition at line 929 of file Image\_ColorOps.C.

References Image< T >::beginw(), Dims::h(), NO\_INIT, and Dims::w().

Referenced by FrameGrabber::grab2rgb().

#### 11.168.2.13 **template<class T> Image<PixRGB<T> > fromYUV411 (const T \* data, const Dims & dims)**

Build an RGB image from a YUV411 C array.

Definition at line 791 of file Image\_ColorOps.C.

References Image< T >::beginw(), Image< T >::endw(), and NO\_INIT.

Referenced by FrameGrabber::grab2rgb().

#### 11.168.2.14 **template<class T> Image<PixRGB<T> > fromYUV411bs (const T \* data, const Dims & dims)**

Build an RGB image from a YUV411 C array, byte-swapping in the process.

Definition at line 812 of file Image\_ColorOps.C.

References Image< T >::beginw(), Image< T >::endw(), and NO\_INIT.

Referenced by FrameGrabber::grab2rgb().

**11.168.2.15    `template<class T> Image<PixRGB<T> > fromYUV411P (const T * data, const Dims & dims)`**

Build an RGB image from a YUV411-planar C array.

Definition at line 876 of file Image\_ColorOps.C.

References Image< T >::beginw(), Dims::h(), NO\_INIT, and Dims::w().

Referenced by FrameGrabber::grab2rgb().

**11.168.2.16    `template<class T> Image<PixRGB<T> > fromYUV420P (const T * data, const Dims & dims)`**

Build an RGB image from a YUV420-planar C array.

Definition at line 900 of file Image\_ColorOps.C.

References Image< T >::beginw(), Dims::h(), NO\_INIT, and Dims::w().

Referenced by FrameGrabber::grab2rgb().

**11.168.2.17    `template<class T> Image<PixRGB<T> > fromYUV422 (const T * data, const Dims & dims)`**

Build an RGB image from a YUV422 C array.

Definition at line 704 of file Image\_ColorOps.C.

References Image< T >::beginw(), Image< T >::endw(), and NO\_INIT.

Referenced by FrameGrabber::grab2rgb().

**11.168.2.18    `template<class T> Image<PixRGB<T> > fromYUV422bs (const T * data, const Dims & dims)`**

Build an RGB image from a YUV422 C array, byte-swapping in the process.

Definition at line 772 of file Image\_ColorOps.C.

References Image< T >::beginw(), Image< T >::endw(), and NO\_INIT.

Referenced by FrameGrabber::grab2rgb().

**11.168.2.19    `template<class T> Image<PixRGB<T> > fromYUV422P (const T * data, const Dims & dims)`**

Build an RGB image from a YUV422-planar C array.

Definition at line 854 of file Image\_ColorOps.C.

References Image< T >::beginw(), Dims::h(), NO\_INIT, and Dims::w().

Referenced by FrameGrabber::grab2rgb().

**11.168.2.20** `template<class T> Image<PixRGB<T> > fromYUV444 (const T * data, const Dims & dims)`

Build an RGB image from a YUV444 C array.

Definition at line 666 of file Image\_ColorOps.C.

References Image< T >::beginw(), Image< T >::endw(), and NO\_INIT.

Referenced by FrameGrabber::grab2rgb().

**11.168.2.21** `template<class T> Image<PixRGB<T> > fromYUV444bs (const T * data, const Dims & dims)`

Build an RGB image from a YUV444 C array, byte-swapping in the process.

Definition at line 685 of file Image\_ColorOps.C.

References Image< T >::beginw(), Image< T >::endw(), and NO\_INIT.

Referenced by FrameGrabber::grab2rgb().

**11.168.2.22** `template<class T> Image<PixRGB<T> > fromYUV444P (const T * data, const Dims & dims)`

Build an RGB image from a YUV422-planar C array.

Definition at line 833 of file Image\_ColorOps.C.

References Image< T >::beginw(), Dims::h(), NO\_INIT, and Dims::w().

Referenced by FrameGrabber::grab2rgb().

**11.168.2.23** `template<class T> void getComponents (const Image< PixRGB< T > > & src, Image< T > & red, Image< T > & green, Image< T > & blue)`

Get the red, green and blue components as 3 monochromatic images.

Definition at line 79 of file Image\_ColorOps.C.

References ASSERT, Image< T >::beginw(), and NO\_INIT.

Referenced by RGBConvolvePyrBuilder< T >::build(), RGBConvolvePyrBuilder< T >::build2(), MultiColorBandChannel::doInput(), maxRGB(), normalizeRGB(), p-SNRcolor(), and takeMax().

**11.168.2.24** `template<class T> void getMinMaxC (const Image< PixRGB< T > > & src, T & mi, T & ma)`

Get min and max of the components.

Definition at line 200 of file Image\_ColorOps.C.

References ASSERT.

Referenced by normalizeC().

**11.168.2.25** `template<class T> void getRGBY (const Image< PixRGB< T > > & src, Image< T > & r, Image< T > & g, Image< T > & b, Image< T > & y, const T thresh)`

Compute R, G, B, Y channels from a luminanceNormalized color image.

This version keeps the four channels separate so that correct results can be obtained without requiring promotion even if the input is of type `PixRGB<byte>`

Definition at line 1125 of file Image\_ColorOps.C.

References ASSERT, Image< T >::begin(), Image< T >::beginw(), lum, and NO\_INIT.

Referenced by MultiConvolveChannel::doInput(), ColorChannel::doInput(), SOColorChannel::doInput(), main(), BeobotVisualCortex::masterProcess(), and BeobotVisualCortex::singleCPUprocess().

**11.168.2.26** `template<class T> void getRGBY (const Image< PixRGB< T > > & src, Image< typename promote_trait< T, float >::TP > & rg, Image< typename promote_trait< T, float >::TP > & by, const T thresh)`

Compute R/G and B/Y channels from a luminanceNormalized color image.

This version does automatic promotion on the results, so that negative values can be represented even if the input is of type `PixRGB<byte>`

Definition at line 1026 of file Image\_ColorOps.C.

**11.168.2.27** `template<class T> void getRGBYsimple (const Image< PixRGB< T > > & src, Image< typename promote_trait< T, float >::TP > & rg, Image< typename promote_trait< T, float >::TP > & by, const T thresh)`

Compute R/G and B/Y channels in a simpler way.

Params are as in getRGBY. The computation for the RG and BY maps is:  $RG = (R - G) / \max(R, G, B)$ ;  $BY = (B - \min(R, G)) / \max(R, G, B)$ .

Definition at line 1083 of file Image\_ColorOps.C.

References ASSERT, and NO\_INIT.

Referenced by ColorChannel::doInput(), and main().

**11.168.2.28** `template<class T> void getYIQ (const Image< PixRGB< T >  
> & src, Image< T > & y, Image< T > & i, Image< T > & q)  
[inline]`

Get YIQ color components.

Definition at line 442 of file Image\_ColorOps.C.

References ASSERT, Image< T >::begin(), Image< T >::beginw(), and NO\_INIT.

Referenced by infoMeasure().

**11.168.2.29** `template<class T> void getYUV (const Image< PixRGB< T >  
> & src, Image< T > & y, Image< T > & u, Image< T > & v)  
[inline]`

Get YUV color components from an RGB image.

Definition at line 469 of file Image\_ColorOps.C.

References ASSERT, Image< T >::begin(), Image< T >::beginw(), and NO\_INIT.

Referenced by main().

**11.168.2.30** `Image<float> hueDistance (const Image< PixRGB< byte > > &  
img, float muR, float muG, float sigR, float sigG, float rho)`

distance between hue of image pixels and a given hue

**Parameters:**

- img* The color image to compare
- muR* the R/I coordinate of the target hue in the C.I.E. color space
- muG* the G/I coordinate in C.I.E.
- sigR* the standard deviation of a 2D Gaussian in R/I direction
- sigG* the standard deviation of a 2D Gaussian in G/I direction
- rho* the correlation between the R/I and G/I directions

**Returns:**

a float image coding for how close the hue in img is to the target hue - 1 for right on the mark, 0 for very different

Definition at line 1301 of file Image\_ColorOps.C.

References Image< T >::beginw(), exp(), NO\_INIT, and RGBtoCIE().

Referenced by HueChannel::doInput().

#### 11.168.2.31 **template<class T> Image<float> infoMeasure (const Image< PixRGB< T > > & src, const float eps, const int size)**

Compute color information measure based in infoFFT.

Definition at line 403 of file Image\_ColorOps.C.

References ASSERT, Image< T >::countThresh(), dct(), getYIQ(), NO\_INIT, and Image< T >::setVal().

Referenced by InformationChannel::doInput().

#### 11.168.2.32 **template<class T> Image<T> luminance (const Image< T > & src)**

Compute luminance of a greyscale image (no-op).

This no-op function is provided so that `luminance()` can be applied to template images where it is not known whether the template argument is scalar or `PixRGB`.

Definition at line 244 of file Image\_ColorOps.C.

Referenced by SimulationViewerStd::drawMotionSixPack(), PbmParser::getFloat(), PngParser::getFloat(), PbmParser::getGray(), PngParser::getGray(), InferoTemporal::input(), VisualCortex::input(), main(), BeobotVisualCortex::masterProcess(), and BeobotVisualCortex::singleCPUprocess().

#### 11.168.2.33 **template<class T> Image<T> luminance (const Image< PixRGB< T > > & src)**

Compute luminance of a color image.

Definition at line 225 of file Image\_ColorOps.C.

#### 11.168.2.34 **template<class T> Image<PixRGB<T> > luminanceNormalize (const Image< PixRGB< T > > & src, const T thresh)**

Normalize my values by those in the lum image.



Definition at line 992 of file Image\_ColorOps.C.

References ASSERT, Image< T >::beginw(), and NO\_INIT.

**11.168.2.35** `template<class T> Image<PixRGB<T> > makeRGB (const Image< T > & red, const Image< T > & green, const Image< T > & blue)`

Set the red, green and blue components from 3 monochromatic images.

Definition at line 55 of file Image\_ColorOps.C.

References ASSERT, Image< T >::begin(), Image< T >::beginw(), Image< T >::endw(), Image< T >::getDims(), Image< T >::isSameSize(), and NO\_INIT.

Referenced by RGBConvolvePyrBuilder< T >::build2(), PsychoDisplay::displayText(), SimulationViewerStd::drawMegaCombo(), main(), normalizeRGB(), and takeMax().

**11.168.2.36** `template<class T> Image<T> maxRGB (const Image< PixRGB< T > > & img)`

Return the pixel-wise maximum of the r, g and b component.

Definition at line 1252 of file Image\_ColorOps.C.

References getComponents(), and takeMax().

**11.168.2.37** `template<class T> Image< PixRGB<T> > normalizeRGB (const Image< PixRGB< T > > & img, PixRGB< T > min, PixRGB< T > max)`

Normalize each component of the RGB image separately.

Normalize the red component with min.red() and max.red() etc.

Definition at line 1238 of file Image\_ColorOps.C.

References getComponents(), makeRGB(), and Image< T >::normalize().

**11.168.2.38** `Image<PixRGB<float> > normalizeRGPolar (const Image< float > & src, const float max, const float min)`

This method will normalize an image based upon the maximum value provided to it. This is either max or the absolute value of min. It then creates an image where all negative values are assigned a red pixel value while all postive numbers are assigned a green pixel value NOTE: please use a [PixRGB](#) type with this class otherwise this method makes no sense

**Parameters:**

- max*** The maximum pixel value
- min*** The minimum pixel value
- image*** The image of values (positive and negative) to be processed

Definition at line 105 of file Image\_ColorOps.C.

References Image< T >::begin(), Image< T >::beginw(), Image< T >::endw(), Image< T >::getDims(), and NO\_INIT.

Referenced by CINNIC::viewNeuronTemplate().

#### 11.168.2.39 **Image<PixRGB<float> > overlayStain (const Image< float > & top, const Image< float > & bottom, const float trans, const char channel)**

Overlay an image over another, however, stain the top image.

this method works almost exactly like overlay except that the bottom image is stained (colored) red green or blue depending on whether char channel is set to r,g, or b

Definition at line 160 of file Image\_ColorOps.C.

References ASSERT, Image< T >::begin(), Image< T >::beginw(), Image< T >::endw(), Image< T >::getDims(), Image< T >::isSameSize(), NO\_INIT, PixRGB< float >::setBlue(), PixRGB< float >::setGreen(), and PixRGB< float >::setRed().

Referenced by contourRun::dumpEnergySigmoid().

#### 11.168.2.40 **template<class T> double pSNRcolor (const Image< PixRGB< T > > & img1, const Image< PixRGB< T > > & img2, const Image< float > & weight)**

Compute weighted peak signal-to-noise ratio between two color images.

This is like the other [pSNRcolor\(\)](#) except that it uses the weighted version of [distance\(\)](#) internally

Definition at line 1220 of file Image\_ColorOps.C.

References ASSERT, distance(), and getComponents().

#### 11.168.2.41 **template<class T> double pSNRcolor (const Image< PixRGB< T > > & img1, const Image< PixRGB< T > > & img2)**

Compute peak signal-to-noise ratio between two color images.

For color images, pSNR computed from the average mean-squared-error from the red, green and blue channels

Definition at line 1203 of file Image\_ColorOps.C.

**11.168.2.42** `void RGBtoCIE (const PixRGB< byte > & rgbColor, float & cr, float & cg, float & intens)`

convert RGB color values into C.I.E. coordinates

**Parameters:**

*rgbColor* the color to convert from

*cr* returns the R/I value

*cg* returns the G/I value

*intens* returns the intensity

Definition at line 1283 of file Image\_ColorOps.C.

References [PixRGB](#)< T >::blue(), [PixRGB](#)< T >::green(), and [PixRGB](#)< T >::red().

Referenced by [HueChannel::HueChannel\(\)](#), and [hueDistance\(\)](#).

**11.168.2.43** `template<class T> Image< PixRGB<T> > setYUV (const Image< T > & y, const Image< T > & u, const Image< T > & v) [inline]`

Build an RGB image from YUV color components.

Definition at line 494 of file Image\_ColorOps.C.

References [ASSERT](#), [Image](#)< T >::begin(), [Image](#)< T >::beginw(), [Image](#)< T >::endw(), and [NO\\_INIT](#).

**11.168.2.44** `template<class T> Image< PixRGB<T> > stain (const Image< T > & src, PixRGB< float > color)`

Create a color image by staining a monochrome image.

Each pixel in the result is given by the stain color multiplied by the corresponding pixel in the monochrome source image. This is entirely equivalent to, but more efficient than, creating a color image from the monochrome image, then multiplying the color image by the stain color.

Definition at line 141 of file Image\_ColorOps.C.

References [Image](#)< T >::begin(), [Image](#)< T >::beginw(), [Image](#)< T >::endw(), [Image](#)< T >::getDims(), and [NO\\_INIT](#).

#### 11.168.2.45 `template<class T> Image< PixRGB<T> > toRGB (const Image< PixRGB< T > > & src)`

Convert RGB image to RGB (no-op).

This no-op function is provided so that `toRGB()` can be applied to template images where it is not known whether the template argument is scalar or `PixRGB`.

Definition at line 265 of file `Image_ColorOps.C`.

Referenced by `PbmParser::getRGB()`, `PfmParser::getRGB()`, `PngParser::getRGB()`, `SimulationViewerCompress::getTraj()`, `SimulationViewerEyeMvt::getTraj()`, `SimulationViewerEyeMvt2::getTraj()`, and `SimulationViewerStd::getTraj()`.

#### 11.168.2.46 `template<class T> Image< PixRGB<T> > toRGB (const Image< T > & src)`

Convert grayscale image to RGB.

Definition at line 249 of file `Image_ColorOps.C`.

#### 11.168.2.47 `void toYUV422 (const Image< PixRGB< byte > > img, byte * y, byte * u, byte * v, const int ystride, const int ustride, const int vstride)`

Fill pre-allocated YUV422 C arrays with values from an image.

Definition at line 724 of file `Image_ColorOps.C`.

References `byte`.

Referenced by `main()`.

#### 11.168.2.48 `template<class T> Image<PixRGB<T> > warp3D (const Image< PixRGB< T > > & ima, const Image< float > & zmap, const float pitch, const float yaw, const float zMax, Dims & dims)`

Warp image in 3D onto z-map.

##### Parameters:

***ima*** Input image

***zmap*** Height map of same size as *ima*

***pitch*** 3D pitch angle, in degrees

***yaw*** 3D yaw angle, in degrees

***dims*** The result will have these dimensions, unless the dimensions are empty (i.e. 0-by-0), in which case the result will be the best size computed from *ima*'s size and the pitch and yaw angles.

Definition at line 270 of file Image\_ColorOps.C.

References ASSERT, CLAMP(), Image< T >::drawLine(), Image< PixRGB< byte > >::getHeight(), Image< T >::getVal(), Image< PixRGB< byte > >::getVal(), Image< PixRGB< byte > >::getWidth(), Dims::h(), Dims::isEmpty(), Image< PixRGB< byte > >::isSameSize(), Image< T >::setVal(), Dims::w(), and ZEROS.

Referenced by main().

## 11.169 Image\_ColorOps.H File Reference

```
#include "saliency.H"
```

```
#include "Promotions.H"
```

```
#include <vector>
```

Include dependency graph for Image\_ColorOps.H:

This graph shows which files directly or indirectly include this file:

### Functions

- `template<class T> Image< PixRGB< T > > makeRGB (const Image< T > &red, const Image< T > &green, const Image< T > &blue)`  
*Set the red, green and blue components from 3 monochromatic images.*
- `template<class T> void getComponents (const Image< PixRGB< T > > &src, Image< T > &red, Image< T > &green, Image< T > &blue)`  
*Get the red, green and blue components as 3 monochromatic images.*
- `Image< PixRGB< float > > normalizeRGPolar (const Image< float > &src, const float max, const float min)`
- `template<class T> Image< PixRGB< T > > stain (const Image< T > &src, PixRGB< float > color)`  
*Create a color image by staining a monochrome image.*
- `Image< PixRGB< float > > overlayStain (const Image< float > &top, const Image< float > &bottom, const float trans, const char channel)`  
*Overlay an image over another, however, stain the top image.*
- `template<class T> void getMinMaxC (const Image< PixRGB< T > > &src, T &mi, T &ma)`  
*Get min and max of the components.*
- `template<class T, class T2> void normalizeC (Image< PixRGB< T > > &src, const T2 nmin, const T2 nmax)`  
*Normalize values between nmin and nmax.*
- `template<class T> Image< T > luminance (const Image< PixRGB< T > > &src)`  
*Compute luminance of a color image.*
- `template<class T> Image< T > luminance (const Image< T > &src)`

*Compute luminance of a greyscale image (no-op).*

- `template<class T> Image< PixRGB< T > > toRGB (const Image< T > &src)`

*Convert grayscale image to RGB.*

- `template<class T> Image< PixRGB< T > > toRGB (const Image< PixRGB< T > > &src)`

*Convert RGB image to RGB (no-op).*

- `template<class T> Image< PixRGB< T > > warp3D (const Image< PixRGB< T > > &ima, const Image< float > &zmap, const float pitch, const float yaw, const float zMax, Dims &dims)`

*Warp image in 3D onto z-map.*

- `template<class T> Image< float > infoMeasure (const Image< PixRGB< T > > &src, const float eps, const int size)`

*Compute color information measure based in infoFFT.*

- `template<class T> void getYIQ (const Image< PixRGB< T > > &src, Image< T > &y, Image< T > &i, Image< T > &q)`

*Get YIQ color components.*

- `template<class T> void getYUV (const Image< PixRGB< T > > &src, Image< T > &y, Image< T > &u, Image< T > &v)`

*Get YUV color components from an RGB image.*

- `template<class T> Image< PixRGB< T > > setYUV (const Image< T > &y, const Image< T > &u, const Image< T > &v)`

*Build an RGB image from YUV color components.*

- `template<class T> Image< PixRGB< T > > fromRGB (const T *data, const Dims &dims)`

*Build an RGB image from an RGB C array.*

- `template<class T> Image< PixRGB< T > > fromARGB (const T *data, const Dims &dims)`

*Build an RGB image from a alpha+RGB C array.*

- `template<class T> Image< PixRGB< T > > fromBGR (const T *data, const Dims &dims)`

*Build an RGB image from a BGR C array.*

- `template<class T> Image< PixRGB< T > > fromBGRA` (const T \*data, const Dims &dims)  
*Build an RGB image from a BGR+alpha C array.*
- `Image< PixRGB< byte > > fromRGB555` (const byte \*data, const Dims &dims)  
*Build an RGB image from an RGB555 C array.*
- `Image< PixRGB< byte > > fromRGB555bs` (const byte \*data, const Dims &dims)  
*Build an RGB image from an RGB555 C array, byte-swapping in the process.*
- `Image< PixRGB< byte > > fromRGB565` (const byte \*data, const Dims &dims)  
*Build an RGB image from an RGB565 C array.*
- `Image< PixRGB< byte > > fromRGB565bs` (const byte \*data, const Dims &dims)  
*Build an RGB image from an RGB565 C array, byte-swapping in the process.*
- `template<class T> Image< PixRGB< T > > fromYUV444` (const T \*data, const Dims &dims)  
*Build an RGB image from a YUV444 C array.*
- `template<class T> Image< PixRGB< T > > fromYUV444bs` (const T \*data, const Dims &dims)  
*Build an RGB image from a YUV444 C array, byte-swapping in the process.*
- `template<class T> Image< PixRGB< T > > fromYUV422` (const T \*data, const Dims &dims)  
*Build an RGB image from a YUV422 C array.*
- `void toYUV422` (const Image< PixRGB< byte > > img, byte \*y, byte \*u, byte \*v, const int ystride=0, const int ustride=0, const int vstride=0)  
*Fill pre-allocated YUV422 C arrays with values from an image.*
- `template<class T> Image< PixRGB< T > > fromYUV422bs` (const T \*data, const Dims &dims)  
*Build an RGB image from a YUV422 C array, byte-swapping in the process.*
- `template<class T> Image< PixRGB< T > > fromYUV411` (const T \*data, const Dims &dims)  
*Build an RGB image from a YUV411 C array.*



- `template<class T> Image< PixRGB< T > > fromYUV411bs` (const T \*data, const Dims &dims)  
*Build an RGB image from a YUV411 C array, byte-swapping in the process.*
- `template<class T> Image< PixRGB< T > > fromYUV444P` (const T \*data, const Dims &dims)  
*Build an RGB image from a YUV422-planar C array.*
- `template<class T> Image< PixRGB< T > > fromYUV422P` (const T \*data, const Dims &dims)  
*Build an RGB image from a YUV422-planar C array.*
- `template<class T> Image< PixRGB< T > > fromYUV411P` (const T \*data, const Dims &dims)  
*Build an RGB image from a YUV411-planar C array.*
- `template<class T> Image< PixRGB< T > > fromYUV420P` (const T \*data, const Dims &dims)  
*Build an RGB image from a YUV420-planar C array.*
- `template<class T> Image< PixRGB< T > > fromYUV410P` (const T \*data, const Dims &dims)  
*Build an RGB image from a YUV410-planar C array.*
- `template<class T> Image< PixRGB< T > > fromMono` (const T \*data, const Dims &dims)  
*Build an RGB image from a grayscale C array of size dims.w()\*dims.h().*
- `template<class T> Image< PixRGB< T > > luminanceNormalize` (const Image< PixRGB< T > > &src, const T thresh)  
*Normalize my values by those in the lum image.*
- `template<class T> void getRGBY` (const Image< PixRGB< T > > &src, Image< typename promote\_trait< T, float >::TP > &rg, Image< typename promote\_trait< T, float >::TP > &by, const T thresh)  
*Compute R/G and B/Y channels from a luminanceNormalized color image.*
- `template<class T> void getRGBYsimple` (const Image< PixRGB< T > > &src, Image< typename promote\_trait< T, float >::TP > &rg, Image< typename promote\_trait< T, float >::TP > &by, const T thresh)  
*Compute R/G and B/Y channels in a simpler way.*

- `template<class T> void getRGBY (const Image< PixRGB< T > > &src, Image< T > &r, Image< T > &g, Image< T > &b, Image< T > &y, const T thresh)`  
*Compute R, G, B, Y channels from a [luminanceNormalized](#) color image.*
- `Image< PixRGB< byte > > contrastModulate (const Image< PixRGB< byte > > &img, const Image< float > &mask, float baseContrast=0.05, byte baseBright=255)`  
*contrast-modulate an RGB image with a mask*
- `template<class T> double pSNRcolor (const Image< PixRGB< T > > &img1, const Image< PixRGB< T > > &img2)`  
*Compute peak signal-to-noise ratio between two color images.*
- `template<class T> double pSNRcolor (const Image< PixRGB< T > > &img1, const Image< PixRGB< T > > &img2, const Image< float > &weight)`  
*Compute weighted peak signal-to-noise ratio between two color images.*
- `template<class T> Image< PixRGB< T > > normalizeRGB (const Image< PixRGB< T > > &img, PixRGB< T > min, PixRGB< T > max)`  
*Normalize each component of the RGB image separately.*
- `template<class T> Image< T > maxRGB (const Image< PixRGB< T > > &img)`  
*Return the pixel-wise maximum of the r, g and b component.*
- `template<class T> Image< PixRGB< T > > colorStain (const Image< T > &src, const T &min, const T &max, const PixRGB< T > &color)`  
*Stain a black and white image with a color.*
- `void RGBtoCIE (const PixRGB< byte > &rgbColor, float &cr, float &cg, float &intens)`  
*convert RGB color values into C.I.E. coordinates*
- `Image< float > hueDistance (const Image< PixRGB< byte > > &img, float muR, float muG, float sigR, float sigG, float rho)`  
*distance between hue of image pixels and a given hue*

### 11.169.1 Detailed Description

Color operations on [Image](#)

Definition in file [Image\\_ColorOps.H](#).

## 11.169.2 Function Documentation

**11.169.2.1** `template<class T> Image< PixRGB<T> > colorStain (const Image< T > & src, const T & min, const T & max, const PixRGB< T > & color)`

Stain a black and white image with a color.

**Parameters:**

*src* the black and white image to be stained  
*min* This value will be converted to 0 (black)  
*max* This value will be converted to color  
*color* The color used for staining

Definition at line 1261 of file Image\_ColorOps.C.

References ASSERT, Image< T >::begin(), Image< T >::beginw(), Image< T >::end(), Image< T >::getDims(), and NO\_INIT.

Referenced by main().

**11.169.2.2** `Image< PixRGB<byte> > contrastModulate (const Image< PixRGB< byte > > & img, const Image< float > & mask, float baseContrast = 0.05, byte baseBright = 255)`

contrast-modulate an RGB image with a mask

Definition at line 1183 of file Image\_ColorOps.C.

References ASSERT, Image< T >::clamp(), and Image< T >::normalize().

Referenced by SimulationViewerStd::getTraj(), and main().

**11.169.2.3** `template<class T> Image<PixRGB<T> > fromARGB (const T * data, const Dims & dims)`

Build an RGB image from an alpha+RGB C array.

Definition at line 528 of file Image\_ColorOps.C.

References Image< T >::beginw(), Image< T >::endw(), and NO\_INIT.

Referenced by FrameGrabber::grab2rgb().

**11.169.2.4** `template<class T> Image<PixRGB<T> > fromBGR (const T * data, const Dims & dims)`

Build an RGB image from a BGR C array.

Definition at line 546 of file Image\_ColorOps.C.

References Image< T >::beginw(), Image< T >::endw(), and NO\_INIT.

Referenced by FrameGrabber::grab2rgb().

#### 11.169.2.5 **template<class T> Image<PixRGB<T> > fromBGRA (const T \* data, const Dims & dims)**

Build an RGB image from a BGR+alpha C array.

Definition at line 564 of file Image\_ColorOps.C.

References Image< T >::beginw(), Image< T >::endw(), and NO\_INIT.

Referenced by FrameGrabber::grab2rgb().

#### 11.169.2.6 **template<class T> Image<PixRGB<T> > fromMono (const T \* data, const Dims & dims)**

Build an RGB image from a grayscale C array of size dims.w()\*dims.h().

Definition at line 975 of file Image\_ColorOps.C.

References Image< T >::beginw(), Image< T >::endw(), and NO\_INIT.

Referenced by FrameGrabber::grab2rgb().

#### 11.169.2.7 **template<class T> Image<PixRGB<T> > fromRGB (const T \* data, const Dims & dims)**

Build an RGB image from an RGB C array.

This will operate a deep copy of the data; this function is somewhat redundant with the Image constructor and is provided for completeness with respect to the fromXXX methods only. Note that data should have 3\*dims.w()\*dims.h() elements, and it is the responsibility of the caller to free that data later.

Definition at line 518 of file Image\_ColorOps.C.

References Image< T >::getArrayPtr(), Dims::h(), NO\_INIT, and Dims::w().

Referenced by FrameGrabber::grab2rgb().

#### 11.169.2.8 **Image<PixRGB<byte> > fromRGB555 (const byte \* data, const Dims & dims)**

Build an RGB image from an RGB555 C array.

Definition at line 581 of file Image\_ColorOps.C.

References Image< T >::beginw(), byte, Image< T >::endw(), and NO\_INIT.

Referenced by FrameGrabber::grab2rgb().

#### 11.169.2.9 Image<PixRGB<byte> > fromRGB555bs (const byte \* data, const Dims & dims)

Build an RGB image from an RGB555 C array, byte-swapping in the process.

Definition at line 602 of file Image\_ColorOps.C.

References Image< T >::beginw(), byte, Image< T >::endw(), and NO\_INIT.

Referenced by FrameGrabber::grab2rgb().

#### 11.169.2.10 Image<PixRGB<byte> > fromRGB565 (const byte \* data, const Dims & dims)

Build an RGB image from an RGB565 C array.

Definition at line 623 of file Image\_ColorOps.C.

References Image< T >::beginw(), byte, Image< T >::endw(), and NO\_INIT.

Referenced by FrameGrabber::grab2rgb().

#### 11.169.2.11 Image<PixRGB<byte> > fromRGB565bs (const byte \* data, const Dims & dims)

Build an RGB image from an RGB565 C array, byte-swapping in the process.

Definition at line 644 of file Image\_ColorOps.C.

References Image< T >::beginw(), byte, Image< T >::endw(), and NO\_INIT.

Referenced by FrameGrabber::grab2rgb().

#### 11.169.2.12 template<class T> Image<PixRGB<T> > fromYUV410P (const T \* data, const Dims & dims)

Build an RGB image from a YUV410-planar C array.

Definition at line 929 of file Image\_ColorOps.C.

References Image< T >::beginw(), Dims::h(), NO\_INIT, and Dims::w().

Referenced by FrameGrabber::grab2rgb().

**11.169.2.13    `template<class T> Image<PixRGB<T> > fromYUV411 (const T * data, const Dims & dims)`**

Build an RGB image from a YUV411 C array.

Definition at line 791 of file Image\_ColorOps.C.

References Image< T >::beginw(), Image< T >::endw(), and NO\_INIT.

Referenced by FrameGrabber::grab2rgb().

**11.169.2.14    `template<class T> Image<PixRGB<T> > fromYUV411bs (const T * data, const Dims & dims)`**

Build an RGB image from a YUV411 C array, byte-swapping in the process.

Definition at line 812 of file Image\_ColorOps.C.

References Image< T >::beginw(), Image< T >::endw(), and NO\_INIT.

Referenced by FrameGrabber::grab2rgb().

**11.169.2.15    `template<class T> Image<PixRGB<T> > fromYUV411P (const T * data, const Dims & dims)`**

Build an RGB image from a YUV411-planar C array.

Definition at line 876 of file Image\_ColorOps.C.

References Image< T >::beginw(), Dims::h(), NO\_INIT, and Dims::w().

Referenced by FrameGrabber::grab2rgb().

**11.169.2.16    `template<class T> Image<PixRGB<T> > fromYUV420P (const T * data, const Dims & dims)`**

Build an RGB image from a YUV420-planar C array.

Definition at line 900 of file Image\_ColorOps.C.

References Image< T >::beginw(), Dims::h(), NO\_INIT, and Dims::w().

Referenced by FrameGrabber::grab2rgb().

**11.169.2.17    `template<class T> Image<PixRGB<T> > fromYUV422 (const T * data, const Dims & dims)`**

Build an RGB image from a YUV422 C array.

Definition at line 704 of file Image\_ColorOps.C.

References Image< T >::beginw(), Image< T >::endw(), and NO\_INIT.

Referenced by FrameGrabber::grab2rgb().

#### 11.169.2.18 `template<class T> Image<PixRGB<T> > fromYUV422bs (const T * data, const Dims & dims)`

Build an RGB image from a YUV422 C array, byte-swapping in the process.

Definition at line 772 of file Image\_ColorOps.C.

References Image< T >::beginw(), Image< T >::endw(), and NO\_INIT.

Referenced by FrameGrabber::grab2rgb().

#### 11.169.2.19 `template<class T> Image<PixRGB<T> > fromYUV422P (const T * data, const Dims & dims)`

Build an RGB image from a YUV422-planar C array.

Definition at line 854 of file Image\_ColorOps.C.

References Image< T >::beginw(), Dims::h(), NO\_INIT, and Dims::w().

Referenced by FrameGrabber::grab2rgb().

#### 11.169.2.20 `template<class T> Image<PixRGB<T> > fromYUV444 (const T * data, const Dims & dims)`

Build an RGB image from a YUV444 C array.

Definition at line 666 of file Image\_ColorOps.C.

References Image< T >::beginw(), Image< T >::endw(), and NO\_INIT.

Referenced by FrameGrabber::grab2rgb().

#### 11.169.2.21 `template<class T> Image<PixRGB<T> > fromYUV444bs (const T * data, const Dims & dims)`

Build an RGB image from a YUV444 C array, byte-swapping in the process.

Definition at line 685 of file Image\_ColorOps.C.

References Image< T >::beginw(), Image< T >::endw(), and NO\_INIT.

Referenced by FrameGrabber::grab2rgb().

### 11.169.2.22 `template<class T> Image<PixRGB<T> > fromYUV444P (const T * data, const Dims & dims)`

Build an RGB image from a YUV422-planar C array.

Definition at line 833 of file Image\_ColorOps.C.

References Image< T >::beginw(), Dims::h(), NO\_INIT, and Dims::w().

Referenced by FrameGrabber::grab2rgb().

### 11.169.2.23 `template<class T> void getComponents (const Image< PixRGB< T > > & src, Image< T > & red, Image< T > & green, Image< T > & blue)`

Get the red, green and blue components as 3 monochromatic images.

Definition at line 79 of file Image\_ColorOps.C.

References ASSERT, Image< T >::beginw(), and NO\_INIT.

Referenced by RGBConvolvePyrBuilder< T >::build(), RGBConvolvePyrBuilder< T >::build2(), MultiColorBandChannel::doInput(), maxRGB(), normalizeRGB(), p-SNRcolor(), and takeMax().

### 11.169.2.24 `template<class T> void getMinMaxC (const Image< PixRGB< T > > & src, T & mi, T & ma)`

Get min and max of the components.

Definition at line 200 of file Image\_ColorOps.C.

References ASSERT.

Referenced by normalizeC().

### 11.169.2.25 `template<class T> void getRGBY (const Image< PixRGB< T > > & src, Image< T > & r, Image< T > & g, Image< T > & b, Image< T > & y, const T thresh)`

Compute R, G, B, Y channels from a luminanceNormalized color image.

This version keeps the four channels separate so that correct results can be obtained without requiring promotion even if the input is of type `PixRGB<byte>`

Definition at line 1125 of file Image\_ColorOps.C.

References ASSERT, Image< T >::begin(), Image< T >::beginw(), lum, and NO\_INIT.



Referenced by SOColorChannel::doInput(), ColorChannel::doInput(), MultiConvolveChannel::doInput(), main(), BeobotVisualCortex::masterProcess(), and BeobotVisualCortex::singleCPUprocess().

**11.169.2.26** `template<class T> void getRGBY (const Image< PixRGB< T > > & src, Image< typename promote_trait< T, float >::TP > & rg, Image< typename promote_trait< T, float >::TP > & by, const T thresh)`

Compute R/G and B/Y channels from a luminanceNormalized color image.

This version does automatic promotion on the results, so that negative values can be represented even if the input is of type `PixRGB<byte>`

Definition at line 1026 of file Image\_ColorOps.C.

References ASSERT, lum, and NO\_INIT.

**11.169.2.27** `template<class T> void getRGBYsimple (const Image< PixRGB< T > > & src, Image< typename promote_trait< T, float >::TP > & rg, Image< typename promote_trait< T, float >::TP > & by, const T thresh)`

Compute R/G and B/Y channels in a simpler way.

Params are as in getRGBY. The computation for the RG and BY maps is:  $RG = (R - G) / \max(R, G, B)$ ;  $BY = (B - \min(R, G)) / \max(R, G, B)$ .

Definition at line 1083 of file Image\_ColorOps.C.

References ASSERT, and NO\_INIT.

Referenced by ColorChannel::doInput(), and main().

**11.169.2.28** `template<class T> void getYIQ (const Image< PixRGB< T > > & src, Image< T > & y, Image< T > & i, Image< T > & q) [inline]`

Get YIQ color components.

Definition at line 442 of file Image\_ColorOps.C.

References ASSERT, Image< T >::begin(), Image< T >::beginw(), and NO\_INIT.

Referenced by infoMeasure().

**11.169.2.29** `template<class T> void getYUV (const Image< PixRGB< T >  
> & src, Image< T > & y, Image< T > & u, Image< T > & v)  
[inline]`

Get YUV color components from an RGB image.

Definition at line 469 of file Image\_ColorOps.C.

References ASSERT, Image< T >::begin(), Image< T >::beginw(), and NO\_INIT.

Referenced by main().

**11.169.2.30** `Image<float> hueDistance (const Image< PixRGB< byte > > &  
img, float muR, float muG, float sigR, float sigG, float rho)`

distance between hue of image pixels and a given hue

**Parameters:**

*img* The color image to compare

*muR* the R/I coordinate of the target hue in the C.I.E. color space

*muG* the G/I coordinate in C.I.E.

*sigR* the standard deviation of a 2D Gaussian in R/I direction

*sigG* the standard deviation of a 2D Gaussian in G/I direction

*rho* the correlation between the R/I and G/I directions

**Returns:**

a float image coding for how close the hue in img is to the target hue - 1 for right on the mark, 0 for very different

Definition at line 1301 of file Image\_ColorOps.C.

References Image< T >::beginw(), exp(), NO\_INIT, and RGBtoCIE().

Referenced by HueChannel::doInput().

**11.169.2.31** `template<class T> Image<float> infoMeasure (const Image<  
PixRGB< T > > & src, const float eps, const int size)`

Compute color information measure based in infoFFT.

Definition at line 403 of file Image\_ColorOps.C.

References ASSERT, Image< T >::countThresh(), dct(), getYIQ(), NO\_INIT, and Image< T >::setVal().

Referenced by InformationChannel::doInput().

### 11.169.2.32 `template<class T> Image<T> luminance (const Image< T > & src)`

Compute luminance of a greyscale image (no-op).

This no-op function is provided so that `luminance()` can be applied to template images where it is not known whether the template argument is scalar or `PixRGB`.

Definition at line 244 of file `Image_ColorOps.C`.

Referenced by `SimulationViewerStd::drawMotionSixPack()`, `PngParser::getFloat()`, `PbmParser::getFloat()`, `PngParser::getGray()`, `PbmParser::getGray()`, `VisualCortex::input()`, `InferoTemporal::input()`, `main()`, `BeobotVisualCortex::masterProcess()`, and `BeobotVisualCortex::singleCPUprocess()`.

### 11.169.2.33 `template<class T> Image<T> luminance (const Image< PixRGB< T > > & src)`

Compute luminance of a color image.

Definition at line 225 of file `Image_ColorOps.C`.

References `ASSERT`, `Image< T >::beginw()`, `Image< T >::endw()`, and `NO_INIT`.

### 11.169.2.34 `template<class T> Image<PixRGB<T> > luminanceNormalize (const Image< PixRGB< T > > & src, const T thresh)`

Normalize my values by those in the lum image.

Definition at line 992 of file `Image_ColorOps.C`.

References `ASSERT`, `Image< T >::beginw()`, and `NO_INIT`.

### 11.169.2.35 `template<class T> Image<PixRGB<T> > makeRGB (const Image< T > & red, const Image< T > & green, const Image< T > & blue)`

Set the red, green and blue components from 3 monochromatic images.

Definition at line 55 of file `Image_ColorOps.C`.

References `ASSERT`, `Image< T >::begin()`, `Image< T >::beginw()`, `Image< T >::endw()`, `Image< T >::getDims()`, `Image< T >::isSameSize()`, and `NO_INIT`.

Referenced by `RGBConvolvePyrBuilder< T >::build2()`, `PsychoDisplay::displayText()`, `SimulationViewerStd::drawMegaCombo()`, `main()`, `normalizeRGB()`, and `takeMax()`.

**11.169.2.36** `template<class T> Image<T> maxRGB (const Image< PixRGB< T > > & img)`

Return the pixel-wise maximum of the r, g and b component.

Definition at line 1252 of file `Image_ColorOps.C`.

References `getComponents()`, and `takeMax()`.

**11.169.2.37** `template<class T, class T2> void normalizeC (Image< PixRGB< T > > & src, const T2 nmin, const T2 nmax) [inline]`

Normalize values between *nmin* and *nmax*.

Definition at line 102 of file `Image_ColorOps.H`.

References `ASSERT`, and `getMinMaxC()`.

**11.169.2.38** `template<class T> Image< PixRGB<T> > normalizeRGB (const Image< PixRGB< T > > & img, PixRGB< T > min, PixRGB< T > max)`

Normalize each component of the RGB image separately.

Normalize the red component with `min.red()` and `max.red()` etc.

Definition at line 1238 of file `Image_ColorOps.C`.

References `getComponents()`, `makeRGB()`, and `Image< T >::normalize()`.

**11.169.2.39** `Image<PixRGB<float> > normalizeRGPolar (const Image< float > & src, const float max, const float min)`

This method will normalize an image based upon the maximum value provided to it. This is either *max* or the absolute value of *min*. It then creates an image where all negative values are assigned a red pixel value while all positive numbers are assigned a green pixel value NOTE: please use a [PixRGB](#) type with this class otherwise this method makes no sense

#### Parameters:

***max*** The maximum pixel value

***min*** The minimum pixel value

***image*** The image of values (positive and negative) to be processed

Definition at line 105 of file `Image_ColorOps.C`.

References `Image< T >::begin()`, `Image< T >::beginw()`, `Image< T >::endw()`, `Image< T >::getDims()`, and `NO_INIT`.

Referenced by CINNIC::viewNeuronTemplate().

**11.169.2.40** `Image<PixRGB<float>> overlayStain (const Image< float > & top, const Image< float > & bottom, const float trans, const char channel)`

Overlay an image over another, however, stain the top image.

this method works almost exactly like overlay except that the bottom image is stained (colored) red green or blue depending on whether char channel is set to r,g, or b

Definition at line 160 of file Image\_ColorOps.C.

References ASSERT, Image< T >::begin(), Image< T >::beginw(), Image< T >::endw(), Image< T >::getDims(), Image< T >::isSameSize(), NO\_INIT, PixRGB< float >::setBlue(), PixRGB< float >::setGreen(), and PixRGB< float >::setRed().

Referenced by contourRun::dumpEnergySigmoid().

**11.169.2.41** `template<class T> double pSNRcolor (const Image< PixRGB< T >> & img1, const Image< PixRGB< T >> & img2, const Image< float > & weight)`

Compute weighted peak signal-to-noise ratio between two color images.

This is like the other `pSNRcolor()` except that it uses the weighted version of `distance()` internally

Definition at line 1220 of file Image\_ColorOps.C.

References ASSERT, distance(), and getComponents().

**11.169.2.42** `template<class T> double pSNRcolor (const Image< PixRGB< T >> & img1, const Image< PixRGB< T >> & img2)`

Compute peak signal-to-noise ratio between two color images.

For color images, pSNR computed from the average mean-squared-error from the red, green and blue channels

Definition at line 1203 of file Image\_ColorOps.C.

References ASSERT, distance(), and getComponents().

#### 11.169.2.43 void RGBtoCIE (const **PixRGB**< **byte** > & *rgbColor*, float & *cr*, float & *cg*, float & *intens*)

convert RGB color values into C.I.E. coordinates

##### Parameters:

***rgbColor*** the color to convert from

***cr*** returns the R/I value

***cg*** returns the G/I value

***intens*** returns the intensity

Definition at line 1283 of file Image\_ColorOps.C.

References **PixRGB**< T >::blue(), **PixRGB**< T >::green(), and **PixRGB**< T >::red().

Referenced by HueChannel::HueChannel(), and hueDistance().

#### 11.169.2.44 template<class T> **Image**<**PixRGB**<T> > setYUV (const **Image**< T > & *y*, const **Image**< T > & *u*, const **Image**< T > & *v*) [inline]

Build an RGB image from YUV color components.

Definition at line 494 of file Image\_ColorOps.C.

References ASSERT, **Image**< T >::begin(), **Image**< T >::beginw(), **Image**< T >::endw(), and NO\_INIT.

#### 11.169.2.45 template<class T> **Image**<**PixRGB**<T> > stain (const **Image**< T > & *src*, **PixRGB**< float > *color*)

Create a color image by staining a monochrome image.

Each pixel in the result is given by the stain color multiplied by the corresponding pixel in the monochrome source image. This is entirely equivalent to, but more efficient than, creating a color image from the monochrome image, then multiplying the color image by the stain color.

Definition at line 141 of file Image\_ColorOps.C.

References **Image**< T >::begin(), **Image**< T >::beginw(), **Image**< T >::endw(), **Image**< T >::getDims(), and NO\_INIT.

#### 11.169.2.46 template<class T> **Image**< **PixRGB**<T> > toRGB (const **Image**< **PixRGB**< T > > & *src*)

Convert RGB image to RGB (no-op).

This no-op function is provided so that `toRGB()` can be applied to template images where it is not known whether the template argument is scalar or `PixRGB`.

Definition at line 265 of file `Image_ColorOps.C`.

Referenced by `PngParser::getRGB()`, `PfmParser::getRGB()`, `PbmParser::getRGB()`, `SimulationViewerStd::getTraj()`, `SimulationViewerEyeMvt2::getTraj()`, `SimulationViewerEyeMvt::getTraj()`, and `SimulationViewerCompress::getTraj()`.

**11.169.2.47** `template<class T> Image< PixRGB<T> > toRGB (const Image< T > & src)`

Convert grayscale image to RGB.

Definition at line 249 of file `Image_ColorOps.C`.

References `ASSERT`, `Image< T >::begin()`, `Image< T >::beginw()`, `Image< T >::endw()`, `Image< T >::getDims()`, `Image< T >::initialized()`, and `NO_INIT`.

**11.169.2.48** `void toYUV422 (const Image< PixRGB< byte > > img, byte * y, byte * u, byte * v, const int ystride = 0, const int ustride = 0, const int vstride = 0)`

Fill pre-allocated YUV422 C arrays with values from an image.

Definition at line 724 of file `Image_ColorOps.C`.

References `byte`.

Referenced by `main()`.

**11.169.2.49** `template<class T> Image<PixRGB<T> > warp3D (const Image< PixRGB< T > > & ima, const Image< float > & zmap, const float pitch, const float yaw, const float zMax, Dims & dims)`

Warp image in 3D onto z-map.

#### Parameters:

*ima* Input image

*zmap* Height map of same size as ima

*pitch* 3D pitch angle, in degrees

*yaw* 3D yaw angle, in degrees

*dims* The result will have these dimensions, unless the dimensions are empty (i.e. 0-by-0), in which case the result will be the best size computed from ima's size and the pitch and yaw angles.

Definition at line 270 of file Image\_ColorOps.C.

References ASSERT, CLAMP(), Image< T >::drawLine(), Image< PixRGB< byte > >::getHeight(), Image< PixRGB< byte > >::getVal(), Image< T >::getVal(), Image< PixRGB< byte > >::getWidth(), Dims::h(), Dims::isEmpty(), Image< PixRGB< byte > >::isSameSize(), Image< T >::setVal(), Dims::w(), and ZEROS.

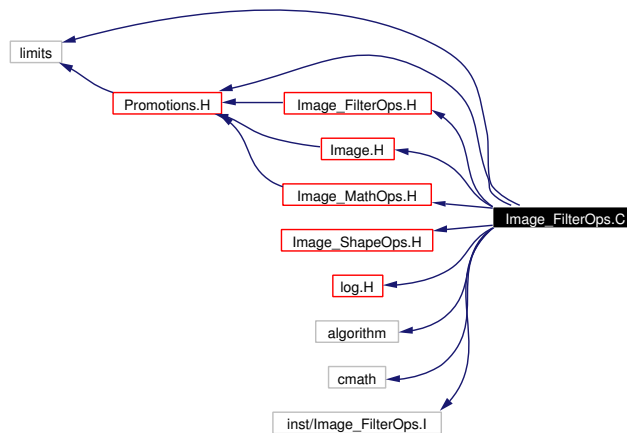
Referenced by main().



## 11.170 Image\_FilterOps.C File Reference

```
#include "Image_FilterOps.H"
#include "Image.H"
#include "Image_ShapeOps.H"
#include "Image_MathOps.H"
#include "log.H"
#include "Promotions.H"
#include <algorithm>
#include <cmath>
#include <limits>
#include "inst/Image_FilterOps.I"
```

Include dependency graph for Image\_FilterOps.C:



## Functions

- `template<class T> Image< typename promote_trait< T, float >::TP > low-Pass3x (const Image< T > &src)`
- `template<class T> Image< typename promote_trait< T, float >::TP > low-Pass3y (const Image< T > &src)`
- `template<class T> Image< typename promote_trait< T, float >::TP > low-Pass5x (const Image< T > &src)`
- `template<class T> Image< typename promote_trait< T, float >::TP > low-Pass5y (const Image< T > &src)`

- `template<class T> Image< typename promote_trait< T, float >::TP > low-Pass5xDecX` (const `Image< T > &src`, const int factor)
  - `template<class T> Image< typename promote_trait< T, float >::TP > low-Pass5yDecY` (const `Image< T > &src`, const int factor)
  - `template<class T> Image< typename promote_trait< T, float >::TP > low-Pass9x` (const `Image< T > &src`)
  - `template<class T> Image< typename promote_trait< T, float >::TP > low-Pass9y` (const `Image< T > &src`)
  - `template<class T> Image< typename promote_trait< T, float >::TP > low-Pass3` (const `Image< T > &src`, const bool go\_x, const bool go\_y)
  - `template<class T> Image< typename promote_trait< T, float >::TP > low-Pass5` (const `Image< T > &src`, const bool go\_x, const bool go\_y)
  - `template<class T> Image< typename promote_trait< T, float >::TP > low-Pass9` (const `Image< T > &src`, const bool go\_x, const bool go\_y)
  - `template<class T> Image< typename promote_trait< T, float >::TP > low-Pass` (const int N, const `Image< T > &src`, const bool go\_x=true, const bool go\_y=true)
  - `template<class T> Image< typename promote_trait< T, float >::TP > low-PassX` (const int N, const `Image< T > &src`)
  - `template<class T> Image< typename promote_trait< T, float >::TP > low-PassY` (const int N, const `Image< T > &src`)
  - `template<class T> Image< typename promote_trait< T, float >::TP > convolve` (const `Image< T > &src`, const `Image< float > &filter`)
  - `template<class T> Image< typename promote_trait< T, float >::TP > convolve` (const `Image< T > &src`, const float \*filter, const int Nx, const int Ny)
  - `template<class T> Image< typename promote_trait< T, float >::TP > convolveClean` (const `Image< T > &src`, const `Image< float > &filter`)
  - `template<class T> Image< typename promote_trait< T, float >::TP > convolveClean` (const `Image< T > &src`, const float \*filter, const int Nx, const int Ny)
  - `template<class T> Image< typename promote_trait< T, float >::TP > convolveCleanZero` (const `Image< T > &src`, const `Image< float > &filter`)
  - `template<class T> Image< typename promote_trait< T, float >::TP > convolveCleanZero` (const `Image< T > &src`, const float \*filter, const int Nx, const int Ny)
  - `template<class T> Image< typename promote_trait< T, float >::TP > opt-Convolve` (const `Image< T > &src`, const `Image< float > &f`)
- This is a somewhat-optimized 2-D convolution.*
- `template<class T> Image< typename promote_trait< T, float >::TP > convolveHmax` (const `Image< T > &src`, const `Image< float > &filter`)
- Brute force, super inefficient 2D convolution (truncated filter boundary).*

- `template<class T> Image< typename promote_trait< T, float >::TP > spatialPoolMax (const Image< T > &src, const int si, const int sj, const int sti, const int stj)`

*Spatial pooling taking the max value over a neighborhood.*

- `template<class T> float featurePoolHmax (const Image< T > &img1, const Image< T > &img2, const Image< T > &img3, const Image< T > &img4, const int si, const int sj, const float s2t)`

*Feature/spatial pooling for the S2 layer of Hmax.*

- `template<class T> Image< typename promote_trait< T, float >::TP > sepFilter (const Image< T > &src, const Image< float > &hFilter, const Image< float > &vFilter)`
- `template<class T> Image< typename promote_trait< T, float >::TP > sepFilter (const Image< T > &src, const float *hFilt, const float *vFilt, const int hfs, const int vfs)`
- `template<class T> Image< typename promote_trait< T, float >::TP > sepFiltClean (const Image< T > &src, const Image< float > &hFilter, const Image< float > &vFilter)`
- `template<class T> Image< typename promote_trait< T, float >::TP > sepFiltClean (const Image< T > &src, const float *hFilter, const float *vFilter, const int hfs, const int vfs)`
- `template<class T> Image< typename promote_trait< T, float >::TP > xFilterClean (const Image< T > &src, const float *hFilt, const int hfs)`
- `template<class T> Image< typename promote_trait< T, float >::TP > yFilterClean (const Image< T > &src, const float *vFilt, const int vfs)`
- `template<class T> Image< typename promote_trait< T, float >::TP > orientedFilter (const Image< T > &src, const float k, const float theta, const float intensity)`
- `template<class T> Image< T > centerSurround (const Image< T > &center, const Image< T > &surround, const bool absol)`
- `template<class T> void centerSurround (const Image< T > &center, const Image< T > &surround, Image< T > &pos, Image< T > &neg)`
- `template<class T> Image< typename promote_trait< T, float >::TP > doubleOpp (const Image< T > &cplus, const Image< T > &cminus, const Image< T > &splus, const Image< T > &sminus)`

*compute double-opponent response*

- `template<class T> void avgOrient (const Image< T > &src, Image< float > &orient, Image< float > &strength)`

*Compute average orientation and strength using steerable filters.*

- `template<class T> Image< T > dogFilter` (const float stddev, const float theta, const int halfsize)  
*Create a DoG kernel; if halfsize is 0, compute it automatically.*
- `template<class T> Image< T > dogFilterHmax` (const float stddev, const float theta, const int cBegin, const int cEnd)  
*Create a dog kernel for HMAX.*
- `template<class T> Image< T > gaborFilter` (const float stddev, const float period, const float phase, const float theta, const T bg, const T ampl)  
*Create a Gabor kernel.*
- `template<class T> Image< T > gaborFilter2` (const float stddev, const float period, const float phase, const float theta, const float sigMod=1.0F, const float amplitude=128.0F)  
*Create a Gabor kernel with default  $1*\sigma^2$  instead of  $2*\sigma^2$ .*
- `Image< float > gaborFilter3` (const float major\_stddev, const float minor\_stddev, const float period, const float phase, const float theta, int size)  
*Produces a Gabor kernel with optionally unequal major+minor axis lengths.*
- `template<class T> Image< T > gaussian2D` (const float stddev, const float sigMod=1.0F, const float amplitude=255.0F)  
*Create a 2D gaussian, compatible with gaborFilter2 for mask purposes.*
- `template<class T> Image< T > gaussianBlob` (const `Dims` &dims, const `Point2D` &center, const float sigmaX, const float sigmaY)  
*Create a 2D Gaussian blob centered at given coords.*
- `double binomial` (int n, int k)
- `Image< float > binomialKernel` (const int sz)  
*Create a 1D binomial distribution kernel (shape of sz-by-1).*
- `template<class T> Image< T > grating` (const int width, const int height, const float period, const float phase, const float theta)  
*Create a sinusoidal grating.*
- `template<class T> Image< T > gaussian` (const float coeff, const float sigma, const int maxhw, const float threshperc)  
*Generate 1D gaussian.*
- `template<class T> Image< T > longRangeExcFilter` (const float factor, const float orient)

*Build a long-range excitatory filter.*

- `template<class T> Image< typename promote_trait< T, float >::TP > conv-Gauss (const Image< T > &src, const float sigmaX, const float sigmaY, const float threshperc)`
- `template<class T> Image< T > fixationMask (const Dims &dims, const Point2D &fixation, const float pixperdeg, const T maxval, const float sigma)`

*Generate a fixation mask, i.e., a Gaussian blob.*

- `template<class T> Image< T > fixationMask (const Image< T > &mask, const Point2D &fixation, const float pixperdeg, const T maxval, const float sigma)`

*Generate a fixation mask, adding it to existing mask.*

- `Image< byte > ones (Dims d)`

*Return an image of Dimensions d filled with 1s.*

- `Image< byte > ones (int w, int h)`

*Return an image of width w and height h filled with 1s.*

- `Image< byte > ones (int w)`

*Return a square image of width w filled with 1s.*

- `template<class T> Image< T > dilateImg (const Image< T > &img, const Image< T > &se, Point2D origin)`

*Dilate the img with structuring element se with a given origin.*

- `template<class T> Image< T > erodeImg (const Image< T > &img, const Image< T > &se, Point2D origin)`

*Erode the img with structuring element se with a given origin.*

- `template<class T> Image< T > openImg (const Image< T > &img, const Image< T > &se, Point2D origin)`

*Open (erode and the close) img with struct element se with a given origin.*

- `template<class T> Image< T > closeImg (const Image< T > &img, const Image< T > &se, Point2D origin)`

*Close (dilate and erode) img with struct element se with a given origin.*

- `template<class T> Image< T > energyNorm (const Image< T > &img)`

*Divide image by the local image energy, then subtract overall mean.*

- `template<class T> Image< T > junctionFilterFull (const Image< T > &i0, const Image< T > &i45, const Image< T > &i90, const Image< T > &i135, const bool r[8], const int dx=6, const int dy=6)`

*Compute response of a junction detector filter.*

- `template<class T> Image< T > junctionFilterPartial (const Image< T > &i0, const Image< T > &i45, const Image< T > &i90, const Image< T > &i135, const bool r[8], const int dx=6, const int dy=6)`

*Compute response of a junction detector filter, partial implementation.*

### 11.170.1 Detailed Description

Filtering operations on [Image](#)

Definition in file [Image\\_FilterOps.C](#).

### 11.170.2 Function Documentation

#### 11.170.2.1 `template<class T> void avgOrient (const Image< T > & src, Image< float > & orient, Image< float > & strength)`

Compute average orientation and strength using steerable filters.

Definition at line 1861 of file [Image\\_FilterOps.C](#).

References [Image](#)< T >::begin(), [Image](#)< T >::beginw(), [Image](#)< T >::getDims(), [Image](#)< T >::getSize(), NO\_INIT, SQR(), and sqrt().

#### 11.170.2.2 `Image<float> binomialKernel (const int sz)`

Create a 1D binomial distribution kernel (shape of sz-by-1).

These kernels are normalized to sum to 1, and are used (implicitly) in the various low-Pass\*() functions.

Definition at line 2156 of file [Image\\_FilterOps.C](#).

References ASSERT, NO\_INIT, and [Image](#)< T >::setVal().

#### 11.170.2.3 `template<class T> Image<T> closeImg (const Image< T > & img, const Image< T > & se, Point2D origin = Point2D(-1,-1))`

Close (dilate and erode) img with struct element se with a given origin.

#### Parameters:

**origin** is the origin of the structuring element. This parameter is optional - the default is the center of the SE

Definition at line 2491 of file Image\_FilterOps.C.

References dilateImg(), and erodeImg().

**11.170.2.4** `template<class T> Image<typename promote_trait<T, float>::TP> convolveHmax (const Image< T > & src, const Image< float > & filter)`

Brute force, super inefficient 2D convolution (truncated filter boundary).

With normalization by the local image energy, a la HMAX

Definition at line 1267 of file Image\_FilterOps.C.

References ASSERT, Image< T >::begin(), Image< T >::beginw(), Image< T >::getHeight(), Image< T >::getWidth(), Image< T >::initialized(), NO\_INIT, and sqrt().

**11.170.2.5** `template<class T> Image<T> dilateImg (const Image< T > & img, const Image< T > & se, Point2D origin = Point2D(-1,-1))`

Dilate the img with structuring element se with a given origin.

**Parameters:**

*origin* is the origin of the structuring element. This parameter is optional - the default is the center of the SE

Definition at line 2329 of file Image\_FilterOps.C.

References ASSERT, Image< T >::begin(), Image< T >::beginw(), Image< T >::getHeight(), Image< T >::getWidth(), Point2D::i, Image< T >::initialized(), Point2D::j, and ZEROS.

Referenced by closeImg(), and openImg().

**11.170.2.6** `template<class T> Image<T> dogFilter (const float stddev, const float theta, const int halfsize)`

Create a DoG kernel; if halfsize is 0, compute it automatically.

Definition at line 1927 of file Image\_FilterOps.C.

References Image< T >::beginw(), exp(), NO\_INIT, and sqrt().

**11.170.2.7** `template<class T> Image<T> dogFilterHmax (const float stddev, const float theta, const int cBegin, const int cEnd)`

Create a dog kernel for HMAX.

Definition at line 1955 of file Image\_FilterOps.C.

References `crop()`, `mean()`, `squared()`, and `sum()`.

**11.170.2.8** `template<class T> Image<typename promote_trait<T, float>::TP> doubleOpp (const Image< T > & cplus, const Image< T > & cminus, const Image< T > & splus, const Image< T > & sminus)`

compute double-opponent response

$(cplus - cminus) [-]$  ( $splus - sminus$ ) where  $[-]$  is subtraction of two images of possibly different sizes, followed by absolute value. The result will have the size of the larger (center) image.

Definition at line 1842 of file Image\_FilterOps.C.

References `ASSERT`, `centerSurround()`, and `Image< T >::isSameSize()`.

**11.170.2.9** `template<class T> Image<T> energyNorm (const Image< T > & img)`

Divide image by the local image energy, then subtract overall mean.

Definition at line 2498 of file Image\_FilterOps.C.

References `lowPass9()`, `mean()`, and `sqrt()`.

Referenced by `buildPyrGabor()`, and `doEnergyNorm()`.

**11.170.2.10** `template<class T> Image<T> erodeImg (const Image< T > & img, const Image< T > & se, Point2D origin = Point2D(-1,-1))`

Erode the `img` with structuring element `se` with a given origin.

#### Parameters:

***origin*** is the origin of the structuring element. This parameter is optional - the default is the center of the SE

Definition at line 2400 of file Image\_FilterOps.C.

References `ASSERT`, `Image< T >::begin()`, `Image< T >::beginw()`, `Image< T >::getHeight()`, `Image< T >::getWidth()`, `Point2D::i`, `Image< T >::initialized()`, `Point2D::j`, and `ZEROS`.

Referenced by `closeImg()`, and `openImg()`.



**11.170.2.11** `template<class T> float featurePoolHmax (const Image< T > & img1, const Image< T > & img2, const Image< T > & img3, const Image< T > & img4, const int si, const int sj, const float s2t)`

Feature/spatial pooling for the S2 layer of [Hmax](#).

Definition at line 1355 of file Image\_FilterOps.C.

References `ASSERT`, `Image< T >::begin()`, `Image< T >::getHeight()`, `Image< T >::getWidth()`, and `Image< T >::isSameSize()`.

**11.170.2.12** `template<class T> Image<T> fixationMask (const Image< T > & mask, const Point2D & fixation, const float pixperdeg, const T maxval, const float sigma)`

Generate a fixation mask, adding it to existing mask.

Definition at line 2289 of file Image\_FilterOps.C.

References `Image< T >::beginw()`, `exp()`, `Image< T >::getHeight()`, `Image< T >::getWidth()`, `Point2D::i`, `Point2D::j`, and `sqrt()`.

Referenced by `fixationMask()`.

**11.170.2.13** `template<class T> Image<T> fixationMask (const Dims & dims, const Point2D & fixation, const float pixperdeg, const T maxval, const float sigma)`

Generate a fixation mask, i.e., a Gaussian blob.

Definition at line 2280 of file Image\_FilterOps.C.

**11.170.2.14** `template<class T> Image<T> gaborFilter (const float stddev, const float period, const float phase, const float theta, const T bg, const T ampl)`

Create a Gabor kernel.

Definition at line 1967 of file Image\_FilterOps.C.

References `Image< T >::beginw()`, `exp()`, `NO_INIT`, `psi()`, and `sqrt()`.

**11.170.2.15** `template<class T> Image<T> gaborFilter2 (const float stddev, const float period, const float phase, const float theta, const float sigMod = 1.0F, const float amplitude = 1.0F)`

Create a Gabor kernel with default  $1 \cdot \sigma^2$  instead of  $2 \cdot \sigma^2$ .

this is essentially used in some psycho metric experiments (for instance , Polat and Sagi, 1994) instead of  $2 \cdot \sigma^2$  so it is included here, if your weird you may make sigMod something other than 1.0 (or 2.0). As always feel free to be loopy

Definition at line 1997 of file Image\_FilterOps.C.

References Image< T >::beginw(), exp(), NO\_INIT, psi(), and sqrt().

#### 11.170.2.16 **Image<float> gaborFilter3 (const float *major\_stddev*, const float *minor\_stddev*, const float *period*, const float *phase*, const float *theta*, int *size* = -1)**

Produces a Gabor kernel with optionally unequal major+minor axis lengths.

##### **Parameters:**

***size*** determines the filter size (in pixels) - in the default case (-1), the size is computed from the stddev

Definition at line 2028 of file Image\_FilterOps.C.

References Image< T >::beginw(), exp(), mean(), NO\_INIT, psi(), and sqrt().

Referenced by buildPyrGabor().

#### 11.170.2.17 **template<class T> Image<T> gaussian (const float *coeff*, const float *sigma*, const int *maxhw*, const float *threshperc* = 1.0)**

Generate 1D gaussian.

##### **Parameters:**

***coeff*** a coefficient so that the peak of the Gaussian has the value coeff. If coeff is given as 0.0, then it will be computed internally as  $1/(\sigma \cdot \sqrt{2 \cdot \pi})$ , i.e., the normal Gaussian coefficient.

***sigma*** the Gaussian standard deviation, in pixels

***maxhw*** maximum allowed filter half-width. If maxhw is 0 there is no limit.

***threshperc*** cut the spatial extent of the kernel so that only values greater than threshperc times the max remain in the kernel. This allows us to automatically compute the appropriate kernel size to use for a given sigma and approximation tolerance given by threshperc.

Definition at line 2202 of file Image\_FilterOps.C.

References exp(), NO\_INIT, Image< T >::setVal(), and sqrt().

**11.170.2.18** `template<class T> Image<T> gaussian2D (const float stddev,  
const float sigMod = 1.0F, const float amplitude = 255.0F)`

Create a 2D gaussian, compatible with gaborFilter2 for mask purposes.

Definition at line 2071 of file Image\_FilterOps.C.

References Image< T >::beginw(), exp(), NO\_INIT, and sqrt().

**11.170.2.19** `template<class T> Image<T> gaussianBlob (const Dims & dims,  
const Point2D & center, const float sigmaX, const float sigmaY)`

Create a 2D Gaussian blob centered at given coords.

Definition at line 2097 of file Image\_FilterOps.C.

References Dims::h(), Point2D::i, Point2D::j, NO\_INIT, Image< T >::setVal(), and Dims::w().

**11.170.2.20** `template<class T> Image<T> grating (const int width, const int  
height, const float period, const float phase, const float theta)`

Create a sinusoidal grating.

Definition at line 2179 of file Image\_FilterOps.C.

References Image< T >::beginw(), NO\_INIT, and psi().

**11.170.2.21** `template<class T> Image<T> junctionFilterFull (const Image< T  
> & i0, const Image< T > & i45, const Image< T > & i90, const  
Image< T > & i135, const bool r[8], const int dx = 6, const int dy =  
6)`

Compute response of a junction detector filter.

In the full implementation here, the junction filter responds only if the relevant features are present and the irrelevant features are absent.

#### Parameters:

*i0* image filtered by a 0deg (horisontal) Gabor filter

*i45* image filtered by a 45deg Gabor filter

*i90* image filtered by a 90deg (vertical) Gabor filter

*i135* image filtered by a 135deg Gabor filter

*r* boolean array of which features are considered relevant to the junction. The first element is for the horizontal (*i0*) feature at (x+dx, y), the second for the 45deg (*i45*) feature at (x+dx, y-dy), and so on, going counterclockwise.

***dx*** horizontal distance from current pixel at which the presence or absence of a given feature should be checked for.

***dy*** vertical distance from current pixel at which the presence or absence of a given feature should be checked for.

Definition at line 2519 of file Image\_FilterOps.C.

References ASSERT, Image< T >::begin(), Image< T >::beginw(), Image< T >::getDims(), Image< T >::getHeight(), Image< T >::getWidth(), Image< T >::isSameSize(), and ZEROS.

Referenced by JunctionChannel::doInput().

**11.170.2.22** `template<class T> Image<T> junctionFilterPartial (const Image< T > & i0, const Image< T > & i45, const Image< T > & i90, const Image< T > & i135, const bool r[8], const int dx = 6, const int dy = 6)`

Compute response of a junction detector filter, partial implementation.

In the partial implementation here, the junction filter responds when the relevant features are present, without consideration of what the values of the irrelevant features might be.

**Parameters:**

***i0*** image filtered by a 0deg (horizontal) Gabor filter

***i45*** image filtered by a 45deg Gabor filter

***i90*** image filtered by a 90deg (vertical) Gabor filter

***i135*** image filtered by a 135deg Gabor filter

***r*** boolean array of which features are considered relevant to the junction. The first element is for the horizontal (*i0*) feature at (x+dx, y), the second for the 45deg (*i45*) feature at (x+dx, y-dy), and so on, going counterclockwise.

***dx*** horizontal distance from current pixel at which the presence or absence of a given feature should be checked for.

***dy*** vertical distance from current pixel at which the presence or absence of a given feature should be checked for.

Definition at line 2629 of file Image\_FilterOps.C.

References ASSERT, Image< T >::begin(), Image< T >::beginw(), Image< T >::getDims(), Image< T >::getHeight(), Image< T >::getWidth(), Image< T >::isSameSize(), and ZEROS.

Referenced by JunctionChannel::doInput().

**11.170.2.23** `template<class T> Image<T> longRangeExcFilter (const float factor, const float orient)`

Build a long-range excitatory filter.

Definition at line 2232 of file Image\_FilterOps.C.

References `Image< T >::beginw()`, `exp()`, `NO_INIT`, and `sqrt()`.

**11.170.2.24** `Image<byte> ones (int w)`

Return a square image of width w filled with 1s.

Definition at line 2322 of file Image\_FilterOps.C.

Referenced by `ones()`.

**11.170.2.25** `Image<byte> ones (int w, int h)`

Return an image of width w and height h filled with 1s.

Definition at line 2316 of file Image\_FilterOps.C.

**11.170.2.26** `Image<byte> ones (Dims d)`

Return an image of Dimensions d filled with 1s.

Definition at line 2308 of file Image\_FilterOps.C.

**11.170.2.27** `template<class T> Image<T> openImg (const Image< T > & img, const Image< T > & se, Point2D origin = Point2D(-1,-1))`

Open (erode and the close) img with struct element se with a given origin.

**Parameters:**

*origin* is the origin of the structuring element. This parameter is optional - the default is the center of the SE

Definition at line 2484 of file Image\_FilterOps.C.

References `dilateImg()`, and `erodeImg()`.

Referenced by `ShapeEstimator::getSmoothMask()`.

**11.170.2.28** `template<class T> Image<typename promote_trait<T,  
float>::TP> optConvolve (const Image< T > & src, const Image<  
float > & filter)`

This is a somewhat-optimized 2-D convolution.

It is optimized in the sense that there are special-case inner loops for (1) cases where the filter doesn't completely overlap with the local image patch because we're at the image boundary (in this case we need a specialized inner loop to handle these boundary conditions), and (2) cases where the filter and local image patch overlap completely (in which case we can use a more efficient inner loop).

Definition at line 1154 of file `Image_FilterOps.C`.

References `ASSERT`, `Image< T >::begin()`, `Image< T >::beginw()`, `Image< T >::getHeight()`, `Image< T >::getWidth()`, `Image< T >::initialized()`, and `NO_INIT`.

Referenced by `buildPyrGabor()`.

**11.170.2.29** `template<class T> Image<typename promote_trait<T,  
float>::TP> spatialPoolMax (const Image< T > & src, const int si,  
const int sj, const int sti, const int stj)`

Spatial pooling taking the max value over a neighborhood.

Compute max over a rectangle of size (*si*, *sj*) starting at positions that increment in steps (*sti*, *stj*)

Definition at line 1318 of file `Image_FilterOps.C`.

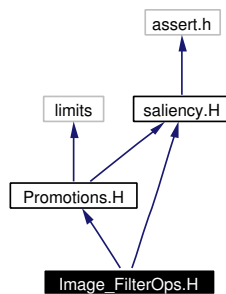
References `Image< T >::begin()`, `Image< T >::beginw()`, `Image< T >::getHeight()`, `Image< T >::getWidth()`, and `NO_INIT`.

## 11.171 Image\_FilterOps.H File Reference

```
#include "saliency.H"
```

```
#include "Promotions.H"
```

Include dependency graph for Image\_FilterOps.H:



This graph shows which files directly or indirectly include this file:

### Functions

- `template<class T_or_RGB> Image< typename promote_trait< T_or_RGB, float >::TP > lowPass3 (const Image< T_or_RGB > &src, const bool go_x=true, const bool go_y=true)`  
*Low-pass filter, coeff \* [0.25 0.5 0.25], applied separately in X and Y.*
- `template<class T_or_RGB> Image< typename promote_trait< T_or_RGB, float >::TP > lowPass3x (const Image< T_or_RGB > &src)`  
*Low-pass filter, coeff \* [0.25 0.5 0.25], applied in X.*
- `template<class T_or_RGB> Image< typename promote_trait< T_or_RGB, float >::TP > lowPass3y (const Image< T_or_RGB > &src)`  
*Low-pass filter, coeff \* [0.25 0.5 0.25], applied in Y.*
- `template<class T_or_RGB> Image< typename promote_trait< T_or_RGB, float >::TP > lowPass5 (const Image< T_or_RGB > &src, const bool go_x=true, const bool go_y=true)`  
*Low-pass filter, Anderson's 5x5 separable.*
- `template<class T_or_RGB> Image< typename promote_trait< T_or_RGB, float >::TP > lowPass5x (const Image< T_or_RGB > &src)`

*Low-pass filter, Anderson's 5x5 separable, applied in X.*

- `template<class T_or_RGB> Image< typename promote_trait< T_or_RGB, float >::TP > lowPass5y (const Image< T_or_RGB > &src)`

*Low-pass filter, Anderson's 5x5 separable, applied in Y.*

- `template<class T_or_RGB> Image< typename promote_trait< T_or_RGB, float >::TP > lowPass5xDecX (const Image< T_or_RGB > &src, const int factor=2)`

*Low-pass filter, Anderson's 5x5 separable, applied in X, and X decimated by factor.*

- `template<class T_or_RGB> Image< typename promote_trait< T_or_RGB, float >::TP > lowPass5yDecY (const Image< T_or_RGB > &src, const int factor=2)`

*Low-pass filter, Anderson's 5x5 separable, applied in Y, and Y decimated by factor.*

- `template<class T_or_RGB> Image< typename promote_trait< T_or_RGB, float >::TP > lowPass9 (const Image< T_or_RGB > &src, const bool go_x=true, const bool go_y=true)`

*Low-pass filter, 9x9 separable.*

- `template<class T_or_RGB> Image< typename promote_trait< T_or_RGB, float >::TP > lowPass9x (const Image< T_or_RGB > &src)`

*Low-pass filter, 9x9 separable, applied in X.*

- `template<class T_or_RGB> Image< typename promote_trait< T_or_RGB, float >::TP > lowPass9y (const Image< T_or_RGB > &src)`

*Low-pass filter, 9x9 separable, applied in Y.*

- `template<class T_or_RGB> Image< typename promote_trait< T_or_RGB, float >::TP > lowPass (const int N, const Image< T_or_RGB > &src, const bool go_x=true, const bool go_y=true)`

*Low-pass filter, NxN separable.*

- `template<class T_or_RGB> Image< typename promote_trait< T_or_RGB, float >::TP > lowPassX (const int N, const Image< T_or_RGB > &src)`

*Low-pass filter, NxN separable, applied in X.*

- `template<class T_or_RGB> Image< typename promote_trait< T_or_RGB, float >::TP > lowPassY (const int N, const Image< T_or_RGB > &src)`

*Low-pass filter, NxN separable, applied in Y.*



- `template<class T_or_RGB> Image< typename promote_trait< T_or_RGB, float >::TP > convolve (const Image< T_or_RGB > &src, const Image< float > &filter)`

*Brute force, super inefficient 2D convolution.*

- `template<class T_or_RGB> Image< typename promote_trait< T_or_RGB, float >::TP > convolve (const Image< T_or_RGB > &src, const float *filter, const int Nx, const int Ny)`

*Brute force, super inefficient 2D convolution, with C filter of Nx\*Ny.*

- `template<class T_or_RGB> Image< typename promote_trait< T_or_RGB, float >::TP > convolveClean (const Image< T_or_RGB > &src, const Image< float > &filter)`

*Brute force, super inefficient 2D convolution (truncated filter boundary).*

- `template<class T_or_RGB> Image< typename promote_trait< T_or_RGB, float >::TP > convolveClean (const Image< T_or_RGB > &src, const float *filter, const int Nx, const int Ny)`

*Brute force, super inefficient 2D convolution, with C filter of Nx\*Ny.*

- `template<class T_or_RGB> Image< typename promote_trait< T_or_RGB, float >::TP > convolveCleanZero (const Image< T_or_RGB > &src, const Image< float > &filter)`

*Brute force, super inefficient 2D convolution (truncated filter boundary).*

- `template<class T_or_RGB> Image< typename promote_trait< T_or_RGB, float >::TP > convolveCleanZero (const Image< T_or_RGB > &src, const float *filter, const int Nx, const int Ny)`

*Brute force, super inefficient 2D convolution, with C filter of Nx\*Ny.*

- `template<class T> Image< typename promote_trait< T, float >::TP > optConvolve (const Image< T > &src, const Image< float > &filter)`

*This is a somewhat-optimized 2-D convolution.*

- `template<class T> Image< typename promote_trait< T, float >::TP > convolveHmax (const Image< T > &src, const Image< float > &filter)`

*Brute force, super inefficient 2D convolution (truncated filter boundary).*

- `template<class T> Image< typename promote_trait< T, float >::TP > spatialPoolMax (const Image< T > &src, const int si, const int sj, const int sti, const int stj)`

*Spatial pooling taking the max value over a neighborhood.*

- `template<class T> float featurePoolHmax (const Image< T > &img1, const Image< T > &img2, const Image< T > &img3, const Image< T > &img4, const int si, const int sj, const float s2t)`

*Feature/spatial pooling for the S2 layer of [Hmax](#).*

- `template<class T_or_RGB> Image< typename promote\_trait< T_or_RGB, float >::TP > sepFilter (const Image< T_or_RGB > &src, const Image< float > &hFilter, const Image< float > &vFilter)`

*Separable filter, any size (size=0 for no filter), zero-fill boundary.*

- `template<class T_or_RGB> Image< typename promote\_trait< T_or_RGB, float >::TP > sepFilter (const Image< T_or_RGB > &src, const float *hFilter, const float *vFilter, const int hfsz, const int vfsz)`
- `template<class T_or_RGB> Image< typename promote\_trait< T_or_RGB, float >::TP > sepFiltClean (const Image< T_or_RGB > &src, const Image< float > &hFilter, const Image< float > &vFilter)`

*Separable filter, any size (size=0 for no filter), clean boundary.*

- `template<class T_or_RGB> Image< typename promote\_trait< T_or_RGB, float >::TP > xFilterClean (const Image< T_or_RGB > &src, const float *hFilter, const int hfsz)`
- `template<class T_or_RGB> Image< typename promote\_trait< T_or_RGB, float >::TP > yFilterClean (const Image< T_or_RGB > &src, const float *vFilter, const int vfsz)`
- `template<class T_or_RGB> Image< typename promote\_trait< T_or_RGB, float >::TP > sepFiltClean (const Image< T_or_RGB > &src, const float *hFilter, const float *vFilter, const int hfsz, const int vfsz)`
- `template<class T_or_RGB> Image< typename promote\_trait< T_or_RGB, float >::TP > orientedFilter (const Image< T_or_RGB > &src, const float k, const float theta, const float intensity=1.0)`

*Oriented filter along theta (deg), spatial freq k.*

- `template<class T_or_RGB> Image< T_or_RGB > centerSurround (const Image< T_or_RGB > &center, const Image< T_or_RGB > &surround, const bool absol=false)`

*Compute center-surround difference, taking the abs if absol==true.*

- `template<class T_or_RGB> void centerSurround (const Image< T_or_RGB > &center, const Image< T_or_RGB > &surround, Image< T_or_RGB > &pos, Image< T_or_RGB > &neg)`

*Compute center-surround difference, without clamping or rectifying.*

- `template<class T> Image< typename promote\_trait< T, float >::TP > doubleOpp (const Image< T > &cplus, const Image< T > &cminus, const Image< T > &splus, const Image< T > &sminus)`

*compute double-opponent response*

- `template<class T> void avgOrient (const Image< T > &src, Image< float > &orient, Image< float > &strength)`

*Compute average orientation and strength using steerable filters.*

- `template<class T> Image< T > dogFilter (const float stddev, const float theta, const int halfsize=0)`

*Create a DoG kernel; if halfsize is 0, compute it automatically.*

- `template<class T> Image< T > dogFilterHmax (const float stddev, const float theta, const int cBegin, const int cEnd)`

*Create a dog kernel for HMAX.*

- `template<class T> Image< T > gaborFilter (const float stddev, const float period, const float phase, const float theta, const T bg=T(0), const T ampl=T(1))`

*Create a Gabor kernel.*

- `template<class T> Image< T > gaborFilter2 (const float stddev, const float period, const float phase, const float theta, const float sigMod=1.0F, const float amplitude=1.0F)`

*Create a Gabor kernel with default  $1*\sigma^2$  instead of  $2*\sigma^2$ .*

- `Image< float > gaborFilter3 (const float major_stddev, const float minor_stddev, const float period, const float phase, const float theta, int size=-1)`

*Produces a Gabor kernel with optionally unequal major+minor axis lengths.*

- `template<class T> Image< T > gaussian2D (const float stddev, const float sigMod=1.0F, const float amplitude=255.0F)`

*Create a 2D gaussian, compatible with gaborFilter2 for mask purposes.*

- `template<class T> Image< T > gaussianBlob (const Dims &dims, const Point2D &center, const float sigmaX, const float sigmaY)`

*Create a 2D Gaussian blob centered at given coords.*

- `Image< float > binomialKernel (const int sz)`

*Create a 1D binomial distribution kernel (shape of sz-by-1).*

- `template<class T> Image< T > grating (const int width, const int height, const float period, const float phase, const float theta)`

*Create a sinusoidal grating.*

- `template<class T> Image< T > gaussian (const float coeff, const float sigma, const int maxhw, const float threshperc=1.0)`

*Generate 1D gaussian.*

- `template<class T> Image< T > longRangeExcFilter` (const float factor, const float orient)

*Build a long-range excitatory filter.*

- `template<class T_or_RGB> Image< typename promote_trait< T_or_RGB, float >::TP > convGauss` (const Image< T\_or\_RGB > &src, const float sigma-X, const float sigmaY, const float threshperc)

*convolve with a separable 2D gaussian*

- `template<class T> Image< T > fixationMask` (const Dims &dims, const Point2D &fixation, const float pixperdeg, const T maxval, const float sigma=0.5f)

*Generate a fixation mask, i.e., a Gaussian blob.*

- `template<class T> Image< T > fixationMask` (const Image< T > &mask, const Point2D &fixation, const float pixperdeg, const T maxval, const float sigma=0.5f)

*Generate a fixation mask, adding it to existing mask.*

- `Image< byte > ones` (Dims d)

*Return an image of Dimensions d filled with 1s.*

- `Image< byte > ones` (int w, int h)

*Return an image of width w and height h filled with 1s.*

- `Image< byte > ones` (int w)

*Return a square image of width w filled with 1s.*

- `template<class T> Image< T > dilateImg` (const Image< T > &img, const Image< T > &se, Point2D origin=Point2D(-1,-1))

*Dilate the img with structuring element se with a given origin.*

- `template<class T> Image< T > erodeImg` (const Image< T > &img, const Image< T > &se, Point2D origin=Point2D(-1,-1))

*Erode the img with structuring element se with a given origin.*

- `template<class T> Image< T > openImg` (const Image< T > &img, const Image< T > &se, Point2D origin=Point2D(-1,-1))

*Open (erode and the close) img with struct element se with a given origin.*

- `template<class T> Image< T > closeImg` (const Image< T > &img, const Image< T > &se, Point2D origin=Point2D(-1,-1))

*Close (dilate and erode) img with struct element se with a given origin.*

- `template<class T> Image< T > energyNorm (const Image< T > &img)`

*Divide image by the local image energy, then subtract overall mean.*

- `template<class T> Image< T > junctionFilterFull (const Image< T > &i0, const Image< T > &i45, const Image< T > &i90, const Image< T > &i135, const bool r[8], const int dx=6, const int dy=6)`

*Compute response of a junction detector filter.*

- `template<class T> Image< T > junctionFilterPartial (const Image< T > &i0, const Image< T > &i45, const Image< T > &i90, const Image< T > &i135, const bool r[8], const int dx=6, const int dy=6)`

*Compute response of a junction detector filter, partial implementation.*

### 11.171.1 Detailed Description

Filtering operations on [Image](#)

Definition in file [Image\\_FilterOps.H](#).

### 11.171.2 Function Documentation

#### 11.171.2.1 `template<class T> void avgOrient (const Image< T > &src, Image< float > &orient, Image< float > &strength)`

Compute average orientation and strength using steerable filters.

Definition at line 1861 of file [Image\\_FilterOps.C](#).

References [Image< T >::begin\(\)](#), [Image< T >::beginw\(\)](#), [Image< T >::getDims\(\)](#), [Image< T >::getSize\(\)](#), [NO\\_INIT](#), [SQR\(\)](#), and [sqrt\(\)](#).

#### 11.171.2.2 `Image<float> binomialKernel (const int sz)`

Create a 1D binomial distribution kernel (shape of sz-by-1).

These kernels are normalized to sum to 1, and are used (implicitly) in the various `low-Pass*()` functions.

Definition at line 2156 of file [Image\\_FilterOps.C](#).

References [ASSERT](#), [NO\\_INIT](#), and [Image< T >::setVal\(\)](#).

**11.171.2.3** `template<class T_or_RGB> void centerSurround (const Image< T_or_RGB > & center, const Image< T_or_RGB > & surround, Image< T_or_RGB > & pos, Image< T_or_RGB > & neg)`

Compute center-surround difference, without clamping or rectifying.

To the (hires) center will be subtracted a (lowres) surround. [Image](#) sizes must be int multiples. This is implemented in a different function from [centerSurround\(\)](#) as it returns two images, for the positive and negative values of the difference

**11.171.2.4** `template<class T_or_RGB> Image<T_or_RGB> centerSurround (const Image< T_or_RGB > & center, const Image< T_or_RGB > & surround, const bool absol = false)`

Compute center-surround difference, taking the abs if absol==true.

To the (hires) center will be subtracted a (lowres) surround. [Image](#) sizes must be int multiples. CAUTION: values will be clamped to the range of your template type.

**Parameters:**

*absol* if true, take the absolute value of the difference (otherwise, negative values are clamped to zero).

**11.171.2.5** `template<class T> Image<T> closeImg (const Image< T > & img, const Image< T > & se, Point2D origin = Point2D(-1,-1))`

Close (dilate and erode) img with struct element se with a given origin.

**Parameters:**

*origin* is the origin of the structuring element. This parameter is optional - the default is the center of the SE

Definition at line 2491 of file Image\_FilterOps.C.

References [dilateImg\(\)](#), and [erodeImg\(\)](#).

**11.171.2.6** `template<class T_or_RGB> Image<typename promote\_trait<T_or_RGB, float>::TP> convGauss (const Image< T_or_RGB > & src, const float sigmaX, const float sigmaY, const float threshperc)`

convolve with a separable 2D gaussian

Referenced by [ScaleSpace::initScaleSpace\(\)](#), and [main\(\)](#).

**11.171.2.7** `template<class T_or_RGB> Image<typename  
promote_trait<T_or_RGB, float>::TP> convolve (const Image<  
T_or_RGB > & src, const float * filter, const int Nx, const int Ny)`

Brute force, super inefficient 2D convolution, with C filter of Nx\*Ny.

Referenced by maxNormalizeFancy().

**11.171.2.8** `template<class T_or_RGB> Image<typename  
promote_trait<T_or_RGB, float>::TP> convolve (const Image<  
T_or_RGB > & src, const Image< float > & filter)`

Brute force, super inefficient 2D convolution.

**11.171.2.9** `template<class T_or_RGB> Image<typename  
promote_trait<T_or_RGB, float>::TP> convolveClean (const  
Image< T_or_RGB > & src, const float * filter, const int Nx, const  
int Ny)`

Brute force, super inefficient 2D convolution, with C filter of Nx\*Ny.

(truncated filter boundary condition)

**11.171.2.10** `template<class T_or_RGB> Image<typename  
promote_trait<T_or_RGB, float>::TP> convolveClean (const  
Image< T_or_RGB > & src, const Image< float > & filter)`

Brute force, super inefficient 2D convolution (truncated filter boundary).

**11.171.2.11** `template<class T_or_RGB> Image<typename  
promote_trait<T_or_RGB, float>::TP> convolveCleanZero (const  
Image< T_or_RGB > & src, const float * filter, const int Nx, const  
int Ny)`

Brute force, super inefficient 2D convolution, with C filter of Nx\*Ny.

(truncated filter boundary condition)

**11.171.2.12** `template<class T_or_RGB> Image<typename  
promote_trait<T_or_RGB, float>::TP> convolveCleanZero (const  
Image< T_or_RGB > & src, const Image< float > & filter)`

Brute force, super inefficient 2D convolution (truncated filter boundary).

check for zero pixels and skip

**11.171.2.13** `template<class T> Image<typename promote_trait<T,  
float>::TP> convolveHmax (const Image< T > & src, const  
Image< float > & filter)`

Brute force, super inefficient 2D convolution (truncated filter boundary).

With normalization by the local image energy, a la HMAX

Definition at line 1267 of file `Image_FilterOps.C`.

References `ASSERT`, `Image< T >::begin()`, `Image< T >::beginw()`, `Image< T >::getHeight()`, `Image< T >::getWidth()`, `Image< T >::initialized()`, `NO_INIT`, and `sqrt()`.

**11.171.2.14** `template<class T> Image<T> dilateImg (const Image< T > &  
img, const Image< T > & se, Point2D origin = Point2D(-1,-1))`

Dilate the img with structuring element *se* with a given origin.

**Parameters:**

*origin* is the origin of the structuring element. This parameter is optional - the default is the center of the SE

Definition at line 2329 of file `Image_FilterOps.C`.

References `ASSERT`, `Image< T >::begin()`, `Image< T >::beginw()`, `Image< T >::getHeight()`, `Image< T >::getWidth()`, `Point2D::i`, `Image< T >::initialized()`, `Point2D::j`, and `ZEROS`.

Referenced by `closeImg()`, and `openImg()`.

**11.171.2.15** `template<class T> Image<T> dogFilter (const float stddev, const  
float theta, const int halfsize = 0)`

Create a DoG kernel; if halfsize is 0, compute it automatically.

Definition at line 1927 of file `Image_FilterOps.C`.

References `Image< T >::beginw()`, `exp()`, `NO_INIT`, and `sqrt()`.

**11.171.2.16** `template<class T> Image<T> dogFilterHmax (const float stddev,  
const float theta, const int cBegin, const int cEnd)`

Create a dog kernel for HMAX.

Definition at line 1955 of file `Image_FilterOps.C`.



References `crop()`, `mean()`, `squared()`, and `sum()`.

**11.171.2.17** `template<class T> Image<typename promote_trait<T, float>::TP> doubleOpp (const Image< T > & cplus, const Image< T > & cminus, const Image< T > & splus, const Image< T > & sminus)`

compute double-opponent response

(*cplus* - *cminus*) [-] (*splus* - *sminus*) where [-] is subtraction of two images of possibly different sizes, followed by absolute value. The result will have the size of the larger (center) image.

Definition at line 1842 of file `Image_FilterOps.C`.

References `ASSERT`, `centerSurround()`, and `Image< T >::isSameSize()`.

**11.171.2.18** `template<class T> Image<T> energyNorm (const Image< T > & img)`

Divide image by the local image energy, then subtract overall mean.

Definition at line 2498 of file `Image_FilterOps.C`.

References `lowPass9()`, `mean()`, and `sqrt()`.

Referenced by `buildPyrGabor()`, and `doEnergyNorm()`.

**11.171.2.19** `template<class T> Image<T> erodeImg (const Image< T > & img, const Image< T > & se, Point2D origin = Point2D(-1,-1))`

Erode the *img* with structuring element *se* with a given origin.

#### Parameters:

***origin*** is the origin of the structuring element. This parameter is optional - the default is the center of the SE

Definition at line 2400 of file `Image_FilterOps.C`.

References `ASSERT`, `Image< T >::begin()`, `Image< T >::beginw()`, `Image< T >::getHeight()`, `Image< T >::getWidth()`, `Point2D::i`, `Image< T >::initialized()`, `Point2D::j`, and `ZEROS`.

Referenced by `closeImg()`, and `openImg()`.

**11.171.2.20** `template<class T> float featurePoolHmax (const Image< T > & img1, const Image< T > & img2, const Image< T > & img3, const Image< T > & img4, const int si, const int sj, const float s2t)`

Feature/spatial pooling for the S2 layer of [Hmax](#).

Definition at line 1355 of file `Image_FilterOps.C`.

References `ASSERT`, `Image< T >::begin()`, `Image< T >::getHeight()`, `Image< T >::getWidth()`, and `Image< T >::isSameSize()`.

**11.171.2.21** `template<class T> Image<T> fixationMask (const Image< T > & mask, const Point2D & fixation, const float pixperdeg, const T maxval, const float sigma = 0.5f)`

Generate a fixation mask, adding it to existing mask.

Definition at line 2289 of file `Image_FilterOps.C`.

References `Image< T >::beginw()`, `exp()`, `Image< T >::getHeight()`, `Image< T >::getWidth()`, `Point2D::i`, `Point2D::j`, and `sqrt()`.

Referenced by `fixationMask()`.

**11.171.2.22** `template<class T> Image<T> fixationMask (const Dims & dims, const Point2D & fixation, const float pixperdeg, const T maxval, const float sigma = 0.5f)`

Generate a fixation mask, i.e., a Gaussian blob.

Definition at line 2280 of file `Image_FilterOps.C`.

References `fixationMask()`, and `ZEROS`.

**11.171.2.23** `template<class T> Image<T> gaborFilter (const float stddev, const float period, const float phase, const float theta, const T bg = T(0), const T ampl = T(1))`

Create a Gabor kernel.

Definition at line 1967 of file `Image_FilterOps.C`.

References `Image< T >::beginw()`, `exp()`, `NO_INIT`, `psi()`, and `sqrt()`.

**11.171.2.24** `template<class T> Image<T> gaborFilter2 (const float stddev,  
const float period, const float phase, const float theta, const float  
sigMod = 1.0F, const float amplitude = 1.0F)`

Create a Gabor kernel with default  $1 \cdot \sigma^2$  instead of  $2 \cdot \sigma^2$ .

this is essentially used in some psycho metric experiments (for instance , Polat and Sagi, 1994) instead of  $2 \cdot \sigma^2$  so it is included here, if your weird you may make sigMod something other than 1.0 (or 2.0). As always feel free to be loopy

Definition at line 1997 of file Image\_FilterOps.C.

References `Image< T >::beginw()`, `exp()`, `NO_INIT`, `psi()`, and `sqrt()`.

**11.171.2.25** `Image<float> gaborFilter3 (const float major_stddev, const float  
minor_stddev, const float period, const float phase, const float theta,  
int size = -1)`

Produces a Gabor kernel with optionally unequal major+minor axis lengths.

**Parameters:**

*size* determines the filter size (in pixels) - in the default case (-1), the size is computed from the stddev

Definition at line 2028 of file Image\_FilterOps.C.

References `Image< T >::beginw()`, `exp()`, `mean()`, `NO_INIT`, `psi()`, and `sqrt()`.

Referenced by `buildPyrGabor()`.

**11.171.2.26** `template<class T> Image<T> gaussian (const float coeff, const  
float sigma, const int maxhw, const float threshperc = 1.0)`

Generate 1D gaussian.

**Parameters:**

*coeff* a coefficient so that the peak of the Gaussian has the value coeff. If coeff is given as 0.0, then it will be computed internally as  $1/(\sigma \cdot \sqrt{2 \cdot \pi})$ , i.e., the normal Gaussian coefficient.

*sigma* the Gaussian standard deviation, in pixels

*maxhw* maximum allowed filter half-width. If maxhw is 0 there is no limit.

*threshperc* cut the spatial extent of the kernel so that only values greater than threshperc times the max remain in the kernel. This allows us to automatically compute the appropriate kernel size to use for a given sigma and approximation tolerance given by threshperc.

Definition at line 2202 of file Image\_FilterOps.C.

References `exp()`, `NO_INIT`, `Image< T >::setVal()`, and `sqrt()`.

**11.171.2.27** `template<class T> Image<T> gaussian2D (const float stddev,  
const float sigMod = 1.0F, const float amplitude = 255.0F)`

Create a 2D gaussian, compatible with `gaborFilter2` for mask purposes.

Definition at line 2071 of file Image\_FilterOps.C.

References `Image< T >::beginw()`, `exp()`, `NO_INIT`, and `sqrt()`.

**11.171.2.28** `template<class T> Image<T> gaussianBlob (const Dims & dims,  
const Point2D & center, const float sigmaX, const float sigmaY)`

Create a 2D Gaussian blob centered at given coords.

Definition at line 2097 of file Image\_FilterOps.C.

References `Dims::h()`, `Point2D::i`, `Point2D::j`, `NO_INIT`, `Image< T >::setVal()`, and `Dims::w()`.

**11.171.2.29** `template<class T> Image<T> grating (const int width, const int  
height, const float period, const float phase, const float theta)`

Create a sinusoidal grating.

Definition at line 2179 of file Image\_FilterOps.C.

References `Image< T >::beginw()`, `NO_INIT`, and `psi()`.

**11.171.2.30** `template<class T> Image<T> junctionFilterFull (const Image< T  
> & i0, const Image< T > & i45, const Image< T > & i90, const  
Image< T > & i135, const bool r[8], const int dx = 6, const int dy =  
6)`

Compute response of a junction detector filter.

In the full implementation here, the junction filter responds only if the relevant features are present and the irrelevant features are absent.

#### Parameters:

*i0* image filtered by a 0deg (horizontal) Gabor filter

*i45* image filtered by a 45deg Gabor filter

*i90* image filtered by a 90deg (vertical) Gabor filter

***i135*** image filtered by a 135deg Gabor filter

***r*** boolean array of which features are considered relevant to the junction. The first element is for the horizontal (*i0*) feature at (*x*+*dx*, *y*), the second for the 45deg (*i45*) feature at (*x*+*dx*, *y*-*dy*), and so on, going counterclockwise.

***dx*** horizontal distance from current pixel at which the presence or absence of a given feature should be checked for.

***dy*** vertical distance from current pixel at which the presence or absence of a given feature should be checked for.

Definition at line 2519 of file Image\_FilterOps.C.

References ASSERT, Image< T >::begin(), Image< T >::beginw(), Image< T >::getDims(), Image< T >::getHeight(), Image< T >::getWidth(), Image< T >::isSameSize(), and ZEROS.

Referenced by JunctionChannel::doInput().

**11.171.2.31** `template<class T> Image<T> junctionFilterPartial (const Image< T > & i0, const Image< T > & i45, const Image< T > & i90, const Image< T > & i135, const bool r[8], const int dx = 6, const int dy = 6)`

Compute response of a junction detector filter, partial implementation.

In the partial implementation here, the junction filter responds when the relevant features are present, without consideration of what the values of the irrelevant features might be.

**Parameters:**

***i0*** image filtered by a 0deg (horizontal) Gabor filter

***i45*** image filtered by a 45deg Gabor filter

***i90*** image filtered by a 90deg (vertical) Gabor filter

***i135*** image filtered by a 135deg Gabor filter

***r*** boolean array of which features are considered relevant to the junction. The first element is for the horizontal (*i0*) feature at (*x*+*dx*, *y*), the second for the 45deg (*i45*) feature at (*x*+*dx*, *y*-*dy*), and so on, going counterclockwise.

***dx*** horizontal distance from current pixel at which the presence or absence of a given feature should be checked for.

***dy*** vertical distance from current pixel at which the presence or absence of a given feature should be checked for.

Definition at line 2629 of file Image\_FilterOps.C.

References ASSERT, Image< T >::begin(), Image< T >::beginw(), Image< T >::getDims(), Image< T >::getHeight(), Image< T >::getWidth(), Image< T >::isSameSize(), and ZEROS.

Referenced by JunctionChannel::doInput().

#### 11.171.2.32 **template<class T> Image<T> longRangeExcFilter (const float factor, const float orient)**

Build a long-range excitatory filter.

Definition at line 2232 of file Image\_FilterOps.C.

References Image< T >::beginw(), exp(), NO\_INIT, and sqrt().

#### 11.171.2.33 **template<class T\_or\_RGB> Image<typename promote\_trait<T\_or\_RGB, float>::TP> lowPass (const int N, const Image< T\_or\_RGB > & src, const bool go\_x = true, const bool go\_y = true)**

Low-pass filter, NxN separable.

Just forwards to lowPass3/lowPass5/lowPass9, but allows the filter width to be selected at runtime.

#### 11.171.2.34 **template<class T\_or\_RGB> Image<typename promote\_trait<T\_or\_RGB, float>::TP> lowPass3 (const Image< T\_or\_RGB > & src, const bool go\_x = true, const bool go\_y = true)**

Low-pass filter, coeff \* [0.25 0.5 0.25], applied separably in X and Y.

Referenced by SimulationViewerEyeMvt::getTraj(), SimulationViewerCompress::getTraj(), main(), and reduce().

#### 11.171.2.35 **template<class T\_or\_RGB> Image<typename promote\_trait<T\_or\_RGB, float>::TP> lowPass3x (const Image< T\_or\_RGB > & src)**

Low-pass filter, coeff \* [0.25 0.5 0.25], applied in X.

#### 11.171.2.36 **template<class T\_or\_RGB> Image<typename promote\_trait<T\_or\_RGB, float>::TP> lowPass3y (const Image< T\_or\_RGB > & src)**

Low-pass filter, coeff \* [0.25 0.5 0.25], applied in Y.

**11.171.2.37** `template<class T_or_RGB> Image<typename  
promote\_trait<T_or_RGB, float>::TP> lowPass5 (const Image<  
T_or_RGB > & src, const bool go_x = true, const bool go_y = true)`

Low-pass filter, Anderson's 5x5 separable.

**11.171.2.38** `template<class T_or_RGB> Image<typename  
promote\_trait<T_or_RGB, float>::TP> lowPass5x (const Image<  
T_or_RGB > & src)`

Low-pass filter, Anderson's 5x5 separable, applied in X.

Referenced by TemplateMatchPyrBuilder::build().

**11.171.2.39** `template<class T_or_RGB> Image<typename  
promote\_trait<T_or_RGB, float>::TP> lowPass5xDecX (const  
Image< T_or_RGB > & src, const int factor = 2)`

Low-pass filter, Anderson's 5x5 separable, applied in X, and X decimated by factor.

**11.171.2.40** `template<class T_or_RGB> Image<typename  
promote\_trait<T_or_RGB, float>::TP> lowPass5y (const Image<  
T_or_RGB > & src)`

Low-pass filter, Anderson's 5x5 separable, applied in Y.

Referenced by TemplateMatchPyrBuilder::build().

**11.171.2.41** `template<class T_or_RGB> Image<typename  
promote\_trait<T_or_RGB, float>::TP> lowPass5yDecY (const  
Image< T_or_RGB > & src, const int factor = 2)`

Low-pass filter, Anderson's 5x5 separable, applied in Y, and Y decimated by factor.

**11.171.2.42** `template<class T_or_RGB> Image<typename  
promote\_trait<T_or_RGB, float>::TP> lowPass9 (const Image<  
T_or_RGB > & src, const bool go_x = true, const bool go_y = true)`

Low-pass filter, 9x9 separable.

Referenced by energyNorm(), VisualBuffer::findMostInterestingTargetLocMax(), and main().

**11.171.2.43** `template<class T_or_RGB> Image<typename promote_trait<T_or_RGB, float>::TP> lowPass9x (const Image<T_or_RGB > & src)`

Low-pass filter, 9x9 separable, applied in X.

**11.171.2.44** `template<class T_or_RGB> Image<typename promote_trait<T_or_RGB, float>::TP> lowPass9y (const Image<T_or_RGB > & src)`

Low-pass filter, 9x9 separable, applied in Y.

**11.171.2.45** `template<class T_or_RGB> Image<typename promote_trait<T_or_RGB, float>::TP> lowPassX (const int N, const Image< T_or_RGB > & src)`

Low-pass filter, NxN separable, applied in X.

Just forwards to lowPass3x/lowPass5x/lowPass9x, but allows the filter width to be selected at runtime.

Referenced by downSize(), and downSizeClean().

**11.171.2.46** `template<class T_or_RGB> Image<typename promote_trait<T_or_RGB, float>::TP> lowPassY (const int N, const Image< T_or_RGB > & src)`

Low-pass filter, NxN separable, applied in Y.

Just forwards to lowPass3y/lowPass5y/lowPass9y, but allows the filter width to be selected at runtime.

Referenced by downSize(), and downSizeClean().

**11.171.2.47** `Image<byte> ones (int w)`

Return a square image of width w filled with 1s.

Definition at line 2322 of file Image\_FilterOps.C.

Referenced by ones().

**11.171.2.48** `Image<byte> ones (int w, int h)`

Return an image of width w and height h filled with 1s.



Definition at line 2316 of file Image\_FilterOps.C.

References ones().

#### 11.171.2.49 Image<byte> ones (Dims d)

Return an image of Dimensions d filled with 1s.

Definition at line 2308 of file Image\_FilterOps.C.

References Image< T >::clear(), and NO\_INIT.

#### 11.171.2.50 template<class T> Image<T> openImg (const Image< T > & img, const Image< T > & se, Point2D origin = Point2D(-1,-1))

Open (erode and the close) img with struct element se with a given origin.

##### Parameters:

*origin* is the origin of the structuring element. This parameter is optional - the default is the center of the SE

Definition at line 2484 of file Image\_FilterOps.C.

References dilateImg(), and erodeImg().

Referenced by ShapeEstimator::getSmoothMask().

#### 11.171.2.51 template<class T> Image<typename promote\_trait<T, float>::TP> optConvolve (const Image< T > & src, const Image< float > & filter)

This is a somewhat-optimized 2-D convolution.

It is optimized in the sense that there are special-case inner loops for (1) cases where the filter doesn't completely overlap with the local image patch because we're at the image boundary (in this case we need a specialized inner loop to handle these boundary conditions), and (2) cases where the filter and local image patch overlap completely (in which case we can use a more efficient inner loop).

Definition at line 1154 of file Image\_FilterOps.C.

References ASSERT, Image< T >::begin(), Image< T >::beginw(), Image< T >::getHeight(), Image< T >::getWidth(), Image< T >::initialized(), and NO\_INIT.

Referenced by buildPyrGabor().

**11.171.2.52** `template<class T_or_RGB> Image<typename promote_trait<T_or_RGB, float>::TP> orientedFilter (const Image< T_or_RGB > & src, const float k, const float theta, const float intensity = 1.0)`

Oriented filter along theta (deg), spatial freq k.

This filter works on the difference between the image and itself low-passed. It modulates this difference by a complex sinusoid, and then low-passes the real and imaginary parts, and finally takes the modulus. This method is by Greenspan et al. (1994). CAUTION: \*you\* must provide as input (i.e., in "this") the difference between an image and itself low-pass filtered. CAUTION: results are possibly bad with the integer versions. Use float.

Referenced by orientedFilterSet().

**11.171.2.53** `template<class T_or_RGB> Image<typename promote_trait<T_or_RGB, float>::TP> sepFiltClean (const Image< T_or_RGB > & src, const Image< float > & hFilter, const Image< float > & vFilter)`

Separable filter, any size (size=0 for no filter), clean boundary.

**11.171.2.54** `template<class T_or_RGB> Image<typename promote_trait<T_or_RGB, float>::TP> sepFilter (const Image< T_or_RGB > & src, const Image< float > & hFilter, const Image< float > & vFilter)`

Separable filter, any size (size=0 for no filter), zero-fill boundary.

**11.171.2.55** `template<class T> Image<typename promote_trait<T, float>::TP> spatialPoolMax (const Image< T > & src, const int si, const int sj, const int sti, const int stj)`

Spatial pooling taking the max value over a neighborhood.

Compute max over a rectangle of size (si, sj) starting at positions that increment in steps (sti, stj)

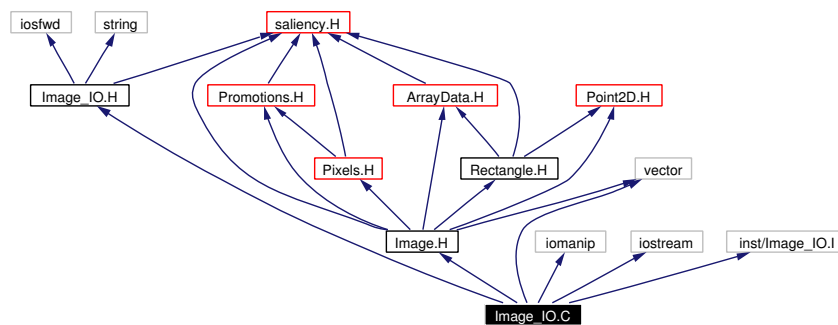
Definition at line 1318 of file Image\_FilterOps.C.

References Image< T >::begin(), Image< T >::beginw(), Image< T >::getHeight(), Image< T >::getWidth(), and NO\_INIT.

## 11.172 Image\_IO.C File Reference

```
#include "Image_IO.H"
#include "Image.H"
#include <iomanip>
#include <iostream>
#include <vector>
#include "inst/Image_IO.I"
```

Include dependency graph for Image\_IO.C:



### Functions

- template<class T> T **printable** (T val)
- int **printable** (byte val)
- template<class T> std::ostream & **operator**<< (std::ostream &os, const **Pix-  
RGB**< T > &pix)
- template<class T> std::ostream & **operator**<< (std::ostream &os, const **Im-  
age**< T > &img)
- void **writeBinImage** (const **Image**< byte > &img, std::ostream &os)  
Write a binary image to "os" in a compact format, storing 5 bits in each character.
- **Image**< byte > **readBinImage** (std::istream &is)  
Read a binary image from "is" that has been written with writeBinImage.
- template<class T> void **writeImageToStream** (std::ostream &os, const **Image**< T > &img)  
Write img to to "os", one line of the image into one line of text.

- `template<class T> void readImageFromStream (std::istream &is, Image< T > &img)`

*Read img from "is" which has been written with `writeImageToStream` before.*

### 11.172.1 Detailed Description

I/O operations for Image's

Definition in file [Image\\_IO.C](#).

### 11.172.2 Function Documentation

#### 11.172.2.1 [Image](#)<byte> `readBinImage` (std::istream & is)

Read a binary image from "is" that has been written with `writeBinImage`.

Definition at line 130 of file [Image\\_IO.C](#).

References `Image`< T >::beginw(), `byte`, `NO_INIT`, and `uint`.

Referenced by `BitObject::readFromStream()`.

#### 11.172.2.2 `template<class T> void readImageFromStream (std::istream & is, Image< T > & img)`

Read img from "is" which has been written with `writeImageToStream` before.

Definition at line 186 of file [Image\\_IO.C](#).

References `Image`< T >::beginw(), `Image`< T >::endw(), `NO_INIT`, and `Image`< T >::resize().

Referenced by `KalmanFilter::readFromStream()`.

#### 11.172.2.3 `void writeBinImage (const Image< byte > & img, std::ostream & os)`

Write a binary image to "os" in a compact format, storing 5 bits in each character.

Definition at line 97 of file [Image\\_IO.C](#).

References `Image`< T >::begin(), `Image`< T >::getHeight(), `Image`< T >::getWidth(), and `uint`.

Referenced by `BitObject::writeToStream()`.

**11.172.2.4    template<class T> void writeImageToStream (std::ostream & os,  
                  const **Image**< T > & *img*)**

Write *img* to to "os", one line of the image into one line of text.

This is only useful for rather small images!

Definition at line 168 of file Image\_IO.C.

References `Image< T >::begin()`, `Image< T >::getDims()`, `Dims::h()`, and `Dims::w()`.

Referenced by `KalmanFilter::writeToStream()`.

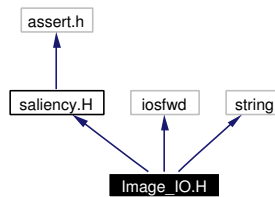
## 11.173 Image\_IO.H File Reference

```
#include "saliency.H"
```

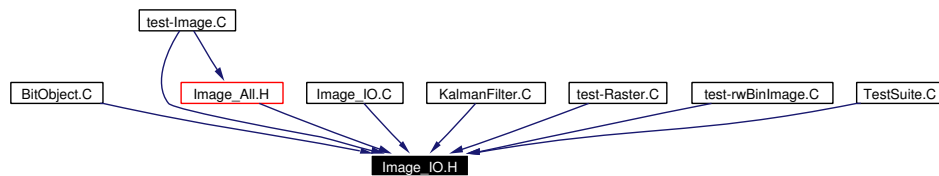
```
#include <iosfwd>
```

```
#include <string>
```

Include dependency graph for Image\_IO.H:



This graph shows which files directly or indirectly include this file:



## Functions

- `template<class T_or_RGB> std::ostream & operator<< (std::ostream &os, const Image< T_or_RGB > &img)`

*Write the pixel values of the Image in a nice human-readable format.*

- `void writeBinImage (const Image< byte > &img, std::ostream &os)`

*Write a binary image to "os" in a compact format, storing 5 bits in each character.*

- `Image< byte > readBinImage (std::istream &is)`

*Read a binary image from "is" that has been written with writeBinImage.*

- `template<class T> void writeImageToStream (std::ostream &os, const Image< T > &img)`

*Write img to to "os", one line of the image into one line of text.*

- `template<class T> void readImageFromStream (std::istream &is, Image< T > &img)`

*Read img from "is" which has been written with writeImageToStream before.*

### 11.173.1 Detailed Description

I/O operations for [Image](#)

Definition in file [Image\\_IO.H](#).

### 11.173.2 Function Documentation

#### 11.173.2.1 `template<class T_or_RGB> std::ostream& operator<< (std::ostream & os, const Image< T_or_RGB > & img)`

Write the pixel values of the [Image](#) in a nice human-readable format.

*Caveat emptor:* it may only be "nice" if the img has a fairly small number of elements, so that one row of the image fits in one row of text. This format is **not** intended to generate a representation that can be re-parsed by the computer to recreate the original image.

#### 11.173.2.2 `Image<byte> readBinImage (std::istream & is)`

Read a binary image from "is" that has been written with writeBinImage.

Definition at line 130 of file Image\_IO.C.

References `Image< T >::beginw()`, `byte`, `NO_INIT`, and `uint`.

Referenced by `BitObject::readFromStream()`.

#### 11.173.2.3 `template<class T> void readImageFromStream (std::istream & is, Image< T > & img)`

Read img from "is" which has been written with writeImageToStream before.

Definition at line 186 of file Image\_IO.C.

References `Image< T >::beginw()`, `Image< T >::endw()`, `NO_INIT`, and `Image< T >::resize()`.

Referenced by `KalmanFilter::readFromStream()`.

**11.173.2.4 void writeBinImage (const Image< byte > & img, std::ostream & os)**

Write a binary image to "os" in a compact format, storing 5 bits in each character.

Definition at line 97 of file Image\_IO.C.

References Image< T >::begin(), Image< T >::getHeight(), Image< T >::getWidth(), and uint.

Referenced by BitObject::writeToStream().

**11.173.2.5 template<class T> void writeImageToStream (std::ostream & os, const Image< T > & img)**

Write img to to "os", one line of the image into one line of text.

This is only useful for rather small images!

Definition at line 168 of file Image\_IO.C.

References Image< T >::begin(), Image< T >::getDims(), Dims::h(), and Dims::w().

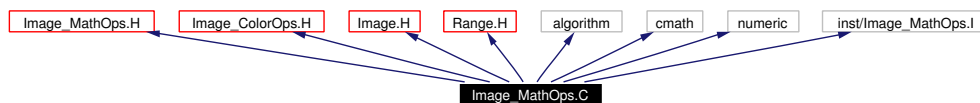
Referenced by KalmanFilter::writeToStream().



## 11.174 Image\_MathOps.C File Reference

```
#include "Image_MathOps.H"
#include "Image_ColorOps.H"
#include "Image.H"
#include "Range.H"
#include <algorithm>
#include <cmath>
#include <numeric>
#include "inst/Image_MathOps.I"
```

Include dependency graph for Image\_MathOps.C:



### Classes

- struct **ToPow**
- struct **EpsInv**

### Functions

- template<class T> bool **operator**== (const [Image](#)< T > &img1, const [Image](#)< T > &img2)
- template<class T> double **sum** (const [Image](#)< T > &a)  
*Returns the sum of all input pixels.*
- template<class T> double **mean** (const [Image](#)< T > &a)  
*Get the average of all input pixels.*
- template<class T> [Range](#)< T > **rangeOf** (const [Image](#)< T > &img)  
*Get the min/max range of the input pixels.*
- template<class T> [Image](#)< T > **remapRange** (const [Image](#)< T > &img, const [Range](#)< T > &from, const [Range](#)< T > &to)  
*Linearly remap pixels from the source to dest dynamic ranges.*

- `Image< byte > normalizeWithScale` (const `Image< float > &src`)  
*Converts a float image to a byte image, normalizing to [0.0-255.0].*
- `template<class T> Image< T > squared` (const `Image< T > &a`)  
*Square each element of the input image.*
- `template<class T> Image< typename promote_trait< T, float >::TP > toPower` (const `Image< T > &a`, double pow)  
*Raise each element of the input image to the given power.*
- `template<class T> Image< T > abs` (const `Image< T > &a`)  
*Take absolute value of every pixel.*
- `template<class T> Image< typename promote_trait< T, float >::TP > hmax-Activation` (const `Image< T > &a`, const float sigma)  
*Compute  $\exp(\text{pixel} / (2 * \sigma^2))$  for every pixel.*
- `template<class T> Image< T > absDiff` (const `Image< T > &b`, const `Image< T > &c`)  
*Returns  $\text{abs}(b - c)$ .*
- `template<class T> Image< PixRGB< T > > absDiff` (const `Image< PixRGB< T > > &b`, const `Image< PixRGB< T > > &c`)  
*Returns  $\text{abs}(b - c)$  for each color channel.*
- `template<class T> Image< T > clampedDiff` (const `Image< T > &b`, const `Image< T > &c`)  
*Returns  $(b - c)$ , clamping to zero all negative values.*
- `template<class T> Image< PixRGB< T > > clampedDiff` (const `Image< PixRGB< T > > &b`, const `Image< PixRGB< T > > &c`)  
*Returns  $(b - c)$  for each color channel, clamping to zero all neg. values.*
- `template<class T> Image< T > binaryReverse` (const `Image< T > &a`, const T val)  
*Returns  $(\text{val} - a)$ .*
- `template<class T> Image< T > average` (const `Image< T > &b`, const `Image< T > &c`)  
*Returns  $(b + c) / 2$ , clean & prevents overflows.*

- `template<class T> Image< T > averageWeighted (Image< T > &a, const Image< T > &b, T *aWeight)`  
Returns  $(a*w + b) / 2$ , the weighted average between images.
- `template<class T> Image< T > takeMax (const Image< T > &a, const Image< T > &b)`  
Returns  $\max(a, b)$ , element by element.
- `template<class T> Image< PixRGB< T > > takeMax (const Image< PixRGB< T > > &a, const Image< PixRGB< T > > &b)`  
Returns  $\max(a, b)$  for each color component separately.
- `template<class T> Image< T > takeMin (const Image< T > &a, const Image< T > &b)`  
Returns  $\min(a, b)$ , element by element.
- `template<class T> Image< T > quadEnergy (const Image< T > &img1, const Image< T > &img2)`
- `template<class T> float RMSerr (const Image< T > &arr1, const Image< T > &arr2)`  
RMS error between arr1 and arr2.
- `template<class T> Image< T > overlay (const Image< T > &top, const Image< T > &bottom, const float trans)`  
Create transparency overlay of two images.
- `template<class T> float sumXY (const Image< T > &img, std::vector< float > &sumX, std::vector< float > &sumY)`  
sum pixels along the X and Y axes (sumX, sumY) and in total (return value)
- `template<class T> double distance (const Image< T > &img1, const Image< T > &img2)`  
Returns Euclidean distance between two images.
- `template<class T> double distance (const Image< T > &img1, const Image< T > &img2, const Image< float > &weight)`  
Returns weighted Euclidean distance between two images.
- `double corrccoef (const Image< float > &img1, const Image< float > &img2)`  
Returns the correlation coefficient  $r^2$  of the two images.
- `template<class T> double corrrpatch (const Image< T > &img1, const Point2D &opleft1, const Dims &patchdims, const Image< T > &img2, const Point2D &opleft2, const double eps)`

*Returns the cross-correlation coefficient  $r$  between two patches.*

- `template<class T> double pSNR (const Image< T > &img1, const Image< T > &img2)`

*Returns peak signal-to-noise ratio.*

- `template<class T> double pSNR (const Image< T > &img1, const Image< T > &img2, const Image< float > &weight)`

*Returns weighted peak signal-to-noise ratio.*

- `template<class T> Image< typename promote_trait< T, float >::TP > sqrt (const Image< T > &a)`

*Take square root of all elements.*

- `template<class T> Image< typename promote_trait< T, float >::TP > inverse (const Image< T > &a, const T eps)`

*Take inverse of all elements  $> eps$  in abs, 0 otherwise.*

- `template<class T> Image< typename promote_trait< T, float >::TP > exp (const Image< T > &a)`

*Take exponential of all pixels.*

- `template<class T> Image< typename promote_trait< T, float >::TP > negexp (const Image< T > &a)`

*Take negative exponential of all pixels, that is  $\exp(-x)$ .*

- `template<class T> bool getCentroidFirstLast (std::vector< T > vect, float &centroid, int &first, int &last)`

*determine the first and last non-zero values and the centroid*

- `template<class T> Point2D centroid (const Image< T > &img, Rectangle &boundingBox, float &cenX, float &cenY)`

*determine the centroid of all the points in the image*

- `template<class T> Point2D centroid (const Image< T > &img)`

*determine the center of mass of all the points in the image*

- `template<class T> Image< T > squash (const Image< T > &ima, const T oldmin, const T newmin, const T oldmid, const T newmid, const T oldmax, const T newmax)`

*Apply a squashing function to the pixel values.*

- `template<class T> Image< T > squash (const Image< T > &src, const T newmin, const T oldmid, const T newmid, const T newmax)`

*Apply a squashing function to the pixel values.*

- `template<class T> Image< T > squash (const Image< T > &src, const T oldmid, const T newmid)`

*Apply a squashing function to the pixel values.*

- `template<class T, class TT> Image< TT > thresholdedMix (const Image< T > &mask, const T &thresh, const Image< TT > &lower, const Image< TT > &higher)`
- `Image< float > matrixMult (const Image< float > &A, const Image< float > &B)`

*Matrix Multiplication:  $C = A * B$ .*

- `Image< float > matrixMult (const Image< float > &A, const Image< float > &B, unsigned int *beginAX, unsigned int *endAX, unsigned int *beginBX, unsigned int *endBX, unsigned int *beginAY, unsigned int *endAY)`

*Matrix Multiplication:  $C = A * B$ .*

- `Image< float > matrixMult (const Image< float > &A, const Image< float > &B, unsigned long *beginAX, unsigned long *endAX, unsigned long *beginBX, unsigned long *endBX, unsigned long *beginAY, unsigned long *endAY)`

*Matrix Multiplication:  $C = A * B$ .*

- `template<class T> std::vector< std::vector< T > > matrixMult (std::vector< std::vector< T > > *A, std::vector< std::vector< T > > *B, unsigned long *beginAX, unsigned long *endAX, unsigned long *beginBX, unsigned long *endBX, unsigned long *beginAY, unsigned long *endAY)`

*Matrix Multiplication:  $C = A * B$ .*

- `template<class T_or_RGB> Image< T_or_RGB > transpose (const Image< T_or_RGB > &M)`

*transpose matrix M*

- `Image< float > eye (const int size)`

*return the identity matrix of dimensions (size x size)*

- `template<class T> Image< T > scramble (const Image< T > &ima)`
- `int32 findMontyMap (const Image< float > &ima, std::vector< Point2D > *coords, float bias=1.0F)`

*take in an image and return it's statistically most relevent points*

- `int32 makeSparseMap (std::vector< Point2D > *coords, std::vector< Point2D > * > *cmap, std::vector< bool > *keep, int inPoints, int outPoints)`

*Take in a vector and decimate it according how many points you want back.*

- template double **sum** (const [Image](#)< double > &a)
- template double **mean** (const [Image](#)< double > &a)
- template [Image](#)< double > **abs** (const [Image](#)< double > &a)
- template [Image](#)< byte > **thresholdedMix** (const [Image](#)< float > &mask, const float &thresh, const [Image](#)< byte > &lower, const [Image](#)< byte > &higher)
- template [Image](#)< double > **toPower** (const [Image](#)< double > &a, double pow)

### 11.174.1 Detailed Description

Mathematical operations on [Image](#)

Definition in file [Image\\_MathOps.C](#).

### 11.174.2 Function Documentation

#### 11.174.2.1 template<class T> [Image](#)<T> **abs** (const [Image](#)< T > &a)

Take absolute value of every pixel.

Definition at line 216 of file [Image\\_MathOps.C](#).

References [Image](#)< T >::begin(), [Image](#)< T >::beginw(), [Image](#)< T >::end(), [Image](#)< T >::getDims(), and NO\_INIT.

Referenced by [Point](#)< T, N >::abs(), [DirectFeedChannel](#)::absSumCoeffs(), [SingleChannel](#)::absSumCoeffs(), [buildPyrGabor](#)(), [Image](#)< T >::lowThreshAbs(), [Point](#)< T, N >::max(), [CameraControl](#)::moveCamXYFrame(), [operator>>\(\)](#), and [GACHromosome](#)::set\_gene().

#### 11.174.2.2 template<class T> [Image](#)< [PixRGB](#)<T> > **absDiff** (const [Image](#)< [PixRGB](#)< T > > &b, const [Image](#)< [PixRGB](#)< T > > &c)

Returns abs(b - c) for each color channel.

Definition at line 265 of file [Image\\_MathOps.C](#).

References ASSERT, [Image](#)< T >::beginw(), [Image](#)< T >::endw(), and NO\_INIT.

Referenced by [ImageCacheAvg](#)< T >::absDiffMean(), [FlickerChannel](#)::doInput(), [TaskRelevanceMapKillStatic](#)::evolve(), and [main](#)().

#### 11.174.2.3 template<class T> [Image](#)<T> **absDiff** (const [Image](#)< T > &b, const [Image](#)< T > &c)

Returns abs(b - c).

Definition at line 243 of file Image\_MathOps.C.

**11.174.2.4** `template<class T> Image<T> average (const Image< T > & b,  
const Image< T > & c)`

Returns  $(b + c) / 2$ , clean & prevents overflows.

Definition at line 464 of file Image\_MathOps.C.

References ASSERT, Image< T >::begin(), Image< T >::beginw(), Image< T >::endw(), Image< T >::getDims(), Image< T >::isSameSize(), and NO\_INIT.

**11.174.2.5** `template<class T> Image<T> averageWeighted (Image< T > & a,  
const Image< T > & b, T * aWeight)`

Returns  $(a*w + b) / 2$ , the weighted average between images.

Definition at line 482 of file Image\_MathOps.C.

References ASSERT, Image< T >::begin(), Image< T >::beginw(), Image< T >::endw(), and Image< T >::isSameSize().

**11.174.2.6** `template<class T> Image<T> binaryReverse (const Image< T > &  
a, const T val)`

Returns  $(val - a)$ .

Definition at line 447 of file Image\_MathOps.C.

References Image< T >::begin(), Image< T >::beginw(), Image< T >::endw(), Image< T >::getDims(), and NO\_INIT.

Referenced by SimulationViewerEyeMvt2::attentionShift(), SimulationViewerEyeMvt2::eyeMovement(), SingleChannel::getRawCSmap(), ShapeEstimator::getSmoothMask(), SimulationViewerCompress::getTraj(), VisualBuffer::inhibit(), TaskRelevanceMapKillN::input(), VisualBuffer::input(), VisualCortex::learn(), and learningCoeff().

**11.174.2.7** `template<class T> Point2D centroid (const Image< T > & img)`

determine the center of mass of all the points in the image

The centroid is rounded to integer pixel positions. When there are no values  $> 0$  in the img, (-1,-1) is returned

Definition at line 904 of file Image\_MathOps.C.

Referenced by VisualBuffer::findMostInterestingTarget(), and main().

#### 11.174.2.8 `template<class T> Point2D centroid (const Image< T > & img, Rectangle & boundingBox, float & cenX, float & cenY)`

determine the centroid of all the points in the image

The centroid is rounded to integer pixel positions. When there are no values > 0 in the img, (-1,-1) is returned

##### Parameters:

***boundingBox*** the bounding box of the object(s) contained in the image is returned

***cenX*** returns the float value of the centroid along the x axis

***cenY*** returns the float value of the centroid along the y axis

##### Returns:

the centroid rounded to integer coordinates

Definition at line 881 of file Image\_MathOps.C.

Referenced by BeobotSensation::getCentroid(), getCentroidFirstLast(), and Beobot::highLevel().

#### 11.174.2.9 `template<class T> Image< PixRGB<T> > clampedDiff (const Image< PixRGB< T > > & b, const Image< PixRGB< T > > & c)`

Returns (b - c) for each color channel, clamping to zero all neg. values.

Definition at line 414 of file Image\_MathOps.C.

#### 11.174.2.10 `template<class T> Image<T> clampedDiff (const Image< T > & b, const Image< T > & c)`

Returns (b - c), clamping to zero all negative values.

Definition at line 334 of file Image\_MathOps.C.

#### 11.174.2.11 `double corrccoef (const Image< float > & img1, const Image< float > & img2)`

Returns the correlation coefficient  $r^2$  of the two images.

Definition at line 676 of file Image\_MathOps.C.

References ASSERT, Image< T >::begin(), Image< T >::end(), Image< T >::getSize(), Image< T >::isSameSize(), and SQR().



**11.174.2.12** `template<class T> double corrpitch (const Image< T > & img1,  
const Point2D & opleft1, const Dims & patchdims, const Image< T  
> & img2, const Point2D & opleft2, const double eps = 1.0e-20)`

Returns the cross-correlation coefficient *r* between two patches.

Note that the computations done here differ from what is done in `corrcoeff`, in that a signed correlation coefficient is returned (between -1 and 1), and we add a convention that if both patches have zero variance then the correlation is 1. If only one patch has zero variance then the correlation is zero. Current implementation requires that the patch fits within both image bounds, and assumes a small patch size so that it will not attempt to use a faster FFT-based implementation which can save time with larger patches.

**Parameters:**

*img1* first image  
*opleft1* coordinates of top-left corner of patch in *img1*  
*patchdims* dimensions of patch  
*img2* second image  
*opleft2* coordinates of top-left corner of patch in *img2*  
*eps* epsilon used to determine whether patch variance is zero

Definition at line 724 of file Image\_MathOps.C.

References `ASSERT`, `Image< T >::begin()`, `Image< T >::getWidth()`, `Dims::h()`, `Point2D::i`, `Point2D::j`, `Image< T >::rectangleOk()`, `sqrt()`, and `Dims::w()`.

Referenced by `TcorrChannel::getOutput()`.

**11.174.2.13** `template<class T> double distance (const Image< T > & img1,  
const Image< T > & img2, const Image< float > & weight)`

Returns weighted Euclidean distance between two images.

Returns  $\sqrt{\sum_{\text{over\_pixels}} (\text{weight}[i] * (\text{img1}[i] - \text{img2}[i])^2)}$

Definition at line 658 of file Image\_MathOps.C.

References `ASSERT`, `Image< T >::begin()`, `Image< T >::end()`, `Image< T >::isSameSize()`, and `sqrt()`.

Referenced by `ImageSpring< T >::computeStiff()`, `VisualCortex::getJetDistMap()`, `ImageSpring< T >::getStatsDist()`, `pSNR()`, and `pSNRcolor()`.

**11.174.2.14** `template<class T> double distance (const Image< T > & img1,  
const Image< T > & img2)`

Returns Euclidean distance between two images.

Returns  $\sqrt{\text{sum\_over\_pixels}((\text{img1}[i] - \text{img2}[i])^2)}$

Definition at line 643 of file Image\_MathOps.C.

**11.174.2.15** `template<class T> Image<typename promote_  
trait<T,float>::TP> exp (const Image< T > &  
a)`

Take exponential of all pixels.

Definition at line 817 of file Image\_MathOps.C.

References Image< T >::begin(), Image< T >::beginw(), Image< T >::end(), Image< T >::getDims(), and NO\_INIT.

Referenced by SingleChannel::centerSurround(), MonkeySaccadeController::computeWhenNewDecision(), VisualObject::createKeys(), distDegrade(), dogFilter(), fixationMask(), gabor2(), gaborFilter(), gaborFilter2(), gaborFilter3(), gaussian(), gaussian2D(), hmaxActivation(), hueDistance(), longRangeExcFilter(), negexp(), MonkeyEyeHeadController::saccade(), CarControl::toSpeedSigmoid(), and SaliencyMap::w().

**11.174.2.16** `Image<float> eye (const int size)`

return the identity matrix of dimensions (size x size)

Definition at line 1150 of file Image\_MathOps.C.

References Image< T >::beginw(), and ZEROS.

Referenced by MonkeySaccadeController::computeWhenNewDecision(), RetinaStd::doGetOutput(), KalmanFilter::init(), and MonkeySaccadeController::MonkeySaccadeController().

**11.174.2.17** `int32 findMontyMap (const Image< float > & ima, std::vector<  
Point2D > * coords, float bias)`

take in an image and return it's statistically most relevant points

Input an image to find a monte carlo like map. This will in essence cut out about half the pixels in a simple quick fashion by applying the formula: select if  $\text{pow}(\text{pixel-Value}, \text{bias}) > \text{minVal} + (\text{maxVal} * \text{rand}() / \text{RAND\_MAX} + 1.0)$  The bias is used to scew the distribution using an exponent if no bias is given, this method will work more efficiently by skipping the exponent computation over each pixel.

#### Parameters:

**ima** This is the input probability map

**coords** this is a pointer to a vector which will hold selected coords

**bais** this screws the probability map by `pow(pixVal,bias)`

Definition at line 1183 of file Image\_MathOps.C.

References `ASSERT`, `Image< PixRGB< byte > >::begin()`, `Image< PixRGB< byte > >::getSize()`, and `Image< PixRGB< byte > >::getWidth()`.

**11.174.2.18** `template<class T> bool getCentroidFirstLast (std::vector< T > vect, float & centroid, int & first, int & last)`

determine the first and last non-zero values and the centroid

**Parameters:**

**vect** a vector of numbers

**centroid** returns the centroid of vect

**first** returns the position of the first non-zero value in vect

**last** returns the position of the last non-zero value in vect

**Returns:**

true if we actually found all these values, false if vect only contains zeros

Definition at line 841 of file Image\_MathOps.C.

References `centroid()`, and `uint`.

Referenced by `centroid()`, and `BitObject::reset()`.

**11.174.2.19** `template<class T> Image<typename promote_trait<T,float>::TP> hmaxActivation (const Image< T > & a, const float sigma)`

Compute  $\exp(\text{pixel} / (2 * \sigma^2))$  for every pixel.

Definition at line 226 of file Image\_MathOps.C.

References `Image< T >::begin()`, `Image< T >::beginw()`, `exp()`, `Image< T >::getDims()`, `Image< T >::getSize()`, and `NO_INIT`.

Referenced by `Hmax::origGetC2()`.

**11.174.2.20** `template<class T> Image<typename promote_trait<T,float>::TP> inverse (const Image< T > & a, const T eps)`

Take inverse of all elements  $> \text{eps}$  in abs, 0 otherwise.

Definition at line 807 of file Image\_MathOps.C.

References Image< T >::begin(), Image< T >::beginw(), Image< T >::end(), Image< T >::getDims(), and NO\_INIT.

Referenced by ImageSpring< T >::getStats().

**11.174.2.21** `int32 makeSparseMap (std::vector< Point2D > * coords, std::vector< Point2D * > * cmap, std::vector< bool > * keep, int inPoints, int outPoints)`

Take in a vector and decimate it according how many points you want back.

In essence this will decimate a list of coordinates attempting to create a list of the size in outPoints. Thus, if you have list of 23890 points and want a list of 300 points, this will take every nth point and create a list of about this size.

**Parameters:**

*coords* This is the list of input coordinates

*cmap* This is the final list of coordinates after decimation

*inPoints* this is the input list size

*outPoints* this is how many points you would like out

Definition at line 1248 of file Image\_MathOps.C.

References ASSERT.

**11.174.2.22** `template<class T> std::vector<std::vector<T> > matrixMult (std::vector< std::vector< T > > * A, std::vector< std::vector< T > > * B, unsigned long * beginAX, unsigned long * endAX, unsigned long * beginBX, unsigned long * endBX, unsigned long * beginAY, unsigned long * endAY)`

Matrix Multiplication:  $C = A * B$ .

The images A and B are interpreted as matrices. For multiplying a (m x n) with a (n x p) matrix, this is the generic matrix multiplication with complexity (m \* n \* p). This version will allow you to multiply values in a matrix that lie between index endA/B and beginA/B. The destination matrix will be the same size as: (endAX - beginAX) x (endAX - beginAX) per usual the width of A (indexed between beginAX and endAX) and height of B MUST be equal. If not then perhaps use transpose... This method is for large matrices

**Parameters:**

*A* input matrix 1

*B* input matrix 2

***beginAX*** starting (width) index in matrix A  
***endAX*** ending (width) index in matrix A  
***beginBY*** starting (height) index in matrix B  
***endBY*** ending (height) index in matrix B  
***beginAY*** starting (height) index in matrix A  
***endAY*** ending (height) index in matrix A

Definition at line 1088 of file Image\_MathOps.C.

References ASSERT, B, and ZEROS.

**11.174.2.23** **Image<float> matrixMult (const Image< float > & A, const Image< float > & B, unsigned long \* *beginAX*, unsigned long \* *endAX*, unsigned long \* *beginBY*, unsigned long \* *endBY*, unsigned long \* *beginAY*, unsigned long \* *endAY*)**

Matrix Multiplication:  $C = A * B$ .

The images A and B are interpreted as matrices. For multiplying a (m x n) with a (n x p) matrix, this is the generic matrix multiplication with complexity (m \* n \* p). This version will allow you to multiply values in a matrix that lie between index endA/B and beginA/B. The destination matrix will be the same size as: (endAX - beginAX) x (endAX - beginAX) per usual the width of A (indexed between beginAX and endAX) and height of B MUST be equal. If not then perhaps use transpose... This method is for large matrices

**Parameters:**

**A** input matrix 1  
**B** input matrix 2  
***beginAX*** starting (width) index in matrix A  
***endAX*** ending (width) index in matrix A  
***beginBY*** starting (height) index in matrix B  
***endBY*** ending (height) index in matrix B  
***beginAY*** starting (height) index in matrix A  
***endAY*** ending (height) index in matrix A

Definition at line 1058 of file Image\_MathOps.C.

**11.174.2.24** **Image<float> matrixMult (const Image< float > & A, const Image< float > & B, unsigned int \* *beginAX*, unsigned int \* *endAX*, unsigned int \* *beginBY*, unsigned int \* *endBY*, unsigned int \* *beginAY*, unsigned int \* *endAY*)**

Matrix Multiplication:  $C = A * B$ .

The images A and B are interpreted as matrices. For multiplying a (m x n) with a (n x p) matrix, this is the generic matrix multiplication with complexity (m \* n \* p). This version will allow you to multiply values in a matrix that lie between index endA/B and beginA/B. The destination matrix will be the same size as: (endAX - beginAX) x (endAX - beginAX) per usual the width of A (indexed between beginAX and endAX) and height of B MUST be equal. If not then perhaps use transpose...

**Parameters:**

**A** input matrix 1  
**B** input matrix 2  
**beginAX** starting (width) index in matrix A  
**endAX** ending (width) index in matrix A  
**beginBY** starting (height) index in matrix B  
**endBY** ending (height) index in matrix B  
**beginAY** starting (height) index in matrix A  
**endAY** ending (height) index in matrix A

Definition at line 1030 of file Image\_MathOps.C.

**11.174.2.25** `Image<float> matrixMult (const Image< float > & A, const Image< float > & B)`

Matrix Multiplication:  $C = A * B$ .

The images A and B are interpreted as matrices. For multiplying a (m x n) with a (n x p) matrix, this is the generic matrix multiplication with complexity (m \* n \* p).

Definition at line 1010 of file Image\_MathOps.C.

**11.174.2.26** `template<class T> double mean (const Image< T > & a)`

Get the average of all input pixels.

Definition at line 107 of file Image\_MathOps.C.

**11.174.2.27** `template<class T> Image<typename promote_ - trait<T,float>::TP> negexp (const Image< T > & a)`

Take negative exponential of all pixels, that is  $\exp(-x)$ .

Definition at line 829 of file Image\_MathOps.C.

References Image< T >::begin(), Image< T >::beginw(), Image< T >::end(), exp(), Image< T >::getDims(), and NO\_INIT.

**11.174.2.28** `Image<byte> normalizeWithScale (const Image<float> & src)`

Converts a float image to a byte image, normalizing to [0.0-255.0].

Two lines of text are appended to the output indicating the original min and max values of the float image.

Definition at line 142 of file Image\_MathOps.C.

References Image< T >::getHeight(), Image< T >::getWidth(), Range< T >::max(), Range< T >::min(), Image< T >::normalize(), Image< T >::paste(), rangeOf(), Image< T >::writeText(), and ZEROS.

Referenced by Raster::WriteFloatNorm().

**11.174.2.29** `template<class T> Image<T> overlay (const Image< T > & top, const Image< T > & bottom, const float trans)`

Create transparency overlay of two images.

by laying top over bottom with transparency of top set to trans. A new image will be returned (use 0 to 100 for percent transparency). NOTE: top and bottom must be the same size! this function only works for grey scale images

Definition at line 587 of file Image\_MathOps.C.

References ASSERT, Image< T >::begin(), Image< T >::beginw(), Image< T >::endw(), Image< T >::getDims(), Image< T >::isSameSize(), and NO\_INIT.

**11.174.2.30** `template<class T> double pSNR (const Image< T > & img1, const Image< T > & img2, const Image<float> & weight)`

Returns weighted peak signal-to-noise ratio.

This is like the other `pSNR()` except that it uses the weighted version of `distance()` internally

Definition at line 785 of file Image\_MathOps.C.

References distance(), and Image< T >::getSize().

**11.174.2.31** `template<class T> double pSNR (const Image< T > & img1, const Image< T > & img2)`

Returns peak signal-to-noise ratio.

pSNR is computed as  $10 \cdot \log_{10}(255^2 / \sigma^2)$  where  $\sigma^2 = 1/N \cdot \sum (x1_i - x2_i)^2$  and is a measure of distortion between two images

Definition at line 776 of file Image\_MathOps.C.

**11.174.2.32    template<class T> [Range](#)<T> rangeOf (const [Image](#)< T > & *img*)**

Get the min/max range of the input pixels.

This provides similar functionality to [Image](#)<T>::getMinMax().

Definition at line 115 of file Image\_MathOps.C.

**11.174.2.33    template<class T> [Image](#)<T> remapRange (const [Image](#)< T > & *img*, const [Range](#)< T > & *from*, const [Range](#)< T > & *to*)**

Linearly remap pixels from the source to dest dynamic ranges.

This is similar to, but more general than, [Image](#)<T>::normalize(). In effect, [Image](#)<T>::normalize() always assumes that the source range is the image's own min/max range. In contrast, [remapRange](#)() allows one to find the global min/max across a range of images, and then use that as the source range while normalizing each the images using that range.

Definition at line 122 of file Image\_MathOps.C.

References [Image](#)< T >::begin(), [Image](#)< T >::beginw(), [Image](#)< T >::endw(), [Image](#)< T >::getDims(), [Range](#)< T >::min(), NO\_INIT, and [Range](#)< T >::range().

**11.174.2.34    template<class T> float RMSerr (const [Image](#)< T > & *arr1*, const [Image](#)< T > & *arr2*)**

RMS error between arr1 and arr2.

Definition at line 570 of file Image\_MathOps.C.

References ASSERT, [Image](#)< T >::begin(), [Image](#)< T >::end(), [Image](#)< T >::getSize(), [Image](#)< T >::initialized(), [Image](#)< T >::isSameSize(), and sqrt().

**11.174.2.35    template<class T> [Image](#)<typename promote\_ -  
[trait](#)<T,float>::TP> sqrt (const [Image](#)< T > & *a*)**

Take square root of all elements.

Definition at line 795 of file Image\_MathOps.C.

**11.174.2.36    template<class T> [Image](#)<T> squared (const [Image](#)< T > & *a*)**

Square each element of the input image.

Definition at line 175 of file Image\_MathOps.C.



References `Image< T >::begin()`, `Image< T >::beginw()`, `Image< T >::end()`, `Image< T >::getDims()`, and `NO_INIT`.

Referenced by `dogFilterHmax()`, `DisparityChannel::doInput()`, and `Hmax::init()`.

**11.174.2.37** `template<class T> Image<T> squash (const Image< T > & src, const T oldmid, const T newmid)`

Apply a squashing function to the pixel values.

In this specialization of the general form of `squash()`, the old min/max are computed using `getMinMax()`, and the new min/max will be kept equal to the old ones, so that there is no overall change in range (unless your mid-point is outside that range).

Definition at line 979 of file `Image_MathOps.C`.

References `Image< T >::getMinMax()`.

Referenced by `VisualBuffer::input()`, and `squash()`.

**11.174.2.38** `template<class T> Image<T> squash (const Image< T > & src, const T newmin, const T oldmid, const T newmid, const T newmax)`

Apply a squashing function to the pixel values.

In this specialization of the general form of `squash()`, the old min/max are computed using `getMinMax()`.

Definition at line 967 of file `Image_MathOps.C`.

**11.174.2.39** `template<class T> Image<T> squash (const Image< T > & src, const T oldmin, const T newmin, const T oldmid, const T newmid, const T oldmax, const T newmax)`

Apply a squashing function to the pixel values.

The squashing function is a 4th-order polynomial with sigmoidal shape. Denoting by  $x$  a pixel value in the original image and by  $y=f(x)$  the transformed pixel value that will be put in the result image, the constraints on the polynomial are: 1)  $f(\text{oldmin})=\text{newmin}$ , 2)  $f(\text{oldmax})=\text{newmax}$ , 3)  $f(\text{oldmid})=\text{newmid}$ , 4)  $f'(\text{oldmin})=0$ , and 5)  $f'(\text{oldmax})=0$ . So, typically, the polynomial will have horizontal slope at both ends of the input range, and will remap the old min/max to the new min/max while also remapping a user-chosen mid-point `oldmid` to `newmid`. Playing with this mid-point allows the user to give more emphasis to higher or lower values while remapping. This is the full form, but two simplified forms are also available that will compute some of the arguments here from the input image. Because the main use of this function is when `oldmin` is the true minimum value of the input image, and `oldmax` its true maximum, here we will not waste time checking for input values outside this range, and their remapped

values are unclear as the polynomial is unconstrained outside the input range. Internal remapping is done using floats, and results are converted back (with possible clamping) to the same type as the original image, so that this function may be efficiently used to do contrast adjustments in [Image<byte>](#) data.

Definition at line 913 of file Image\_MathOps.C.

#### 11.174.2.40 `template<class T> double sum (const Image< T > & a)`

Returns the sum of all input pixels.

Definition at line 66 of file Image\_MathOps.C.

#### 11.174.2.41 `template<class T> float sumXY (const Image< T > & img, std::vector< float > & sumX, std::vector< float > & sumY)`

sum pixels along the X and Y axes (sumX, sumY) and in total (return value)

##### Parameters:

*img* is a w x h size image

*sumX* will return a vector of length w with the sum over the rows of img

*sumY* will return a vector of length h with the sum over the columns

Definition at line 614 of file Image\_MathOps.C.

References [Image< T >::begin\(\)](#), [Image< T >::getHeight\(\)](#), and [Image< T >::getWidth\(\)](#).

Referenced by [centroid\(\)](#), and [BitObject::reset\(\)](#).

#### 11.174.2.42 `template<class T> Image< PixRGB<T> > takeMax (const Image< PixRGB< T > > & a, const Image< PixRGB< T > > & b)`

Returns max(a, b) for each color component separately.

Definition at line 515 of file Image\_MathOps.C.

References [getComponents\(\)](#), and [makeRGB\(\)](#).

Referenced by [SimulationViewerStd::attentionShift\(\)](#), [VisualCortexEyeMvt::evolve\(\)](#), [extractBitObjects\(\)](#), [SimulationViewerStd::eyeMovement\(\)](#), [ImageCacheMinMax< T >::getMax\(\)](#), [VisualCortexEyeMvt::getOutput\(\)](#), [ShapeEstimator::getSmoothMask\(\)](#), [SimulationViewerStd::headMovement\(\)](#), [VisualBuffer::input\(\)](#), [main\(\)](#), [maxRGB\(\)](#), and [showAllObjects\(\)](#).

**11.174.2.43** `template<class T> Image<T> takeMax (const Image< T > & a,  
const Image< T > & b)`

Returns max(a, b), element by element.

Definition at line 494 of file Image\_MathOps.C.

**11.174.2.44** `template<class T> Image<T> takeMin (const Image< T > & a,  
const Image< T > & b)`

Returns min(a, b), element by element.

Definition at line 527 of file Image\_MathOps.C.

References ASSERT, [Image](#)< T >::begin(), [Image](#)< T >::beginw(), [Image](#)< T >::endw(), [Image](#)< T >::getDims(), [Image](#)< T >::isSameSize(), and NO\_INIT.

Referenced by BitObject::doesIntersect(), ImageCacheMinMax< T >::getMin(), and SimulationViewerCompress::getTraj().

**11.174.2.45** `template<class T> Image<typename promote\_-  
trait<T,float>::TP> toPower (const Image< T > & a, double  
pow)`

Raise each element of the input image to the given power.

Definition at line 207 of file Image\_MathOps.C.

References [Image](#)< T >::begin(), [Image](#)< T >::beginw(), [Image](#)< T >::end(), [Image](#)< T >::getDims(), and NO\_INIT.

Referenced by SoxChannel::getNonlinearResponse(), and SurpriseMap< T >::surprise().

**11.174.2.46** `template<class T_or_RGB> Image<T_or_RGB> transpose (const  
Image< T_or_RGB > & M)`

transpose matrix M

Definition at line 1128 of file Image\_MathOps.C.

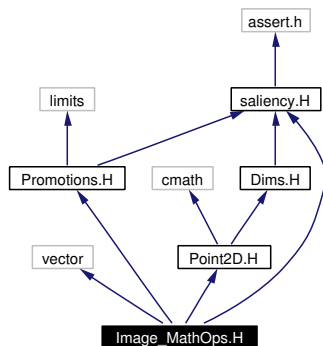
References [Image](#)< T >::begin(), [Image](#)< T >::beginw(), [Image](#)< T >::getHeight(), [Image](#)< T >::getWidth(), and NO\_INIT.

Referenced by KalmanFilter::init().

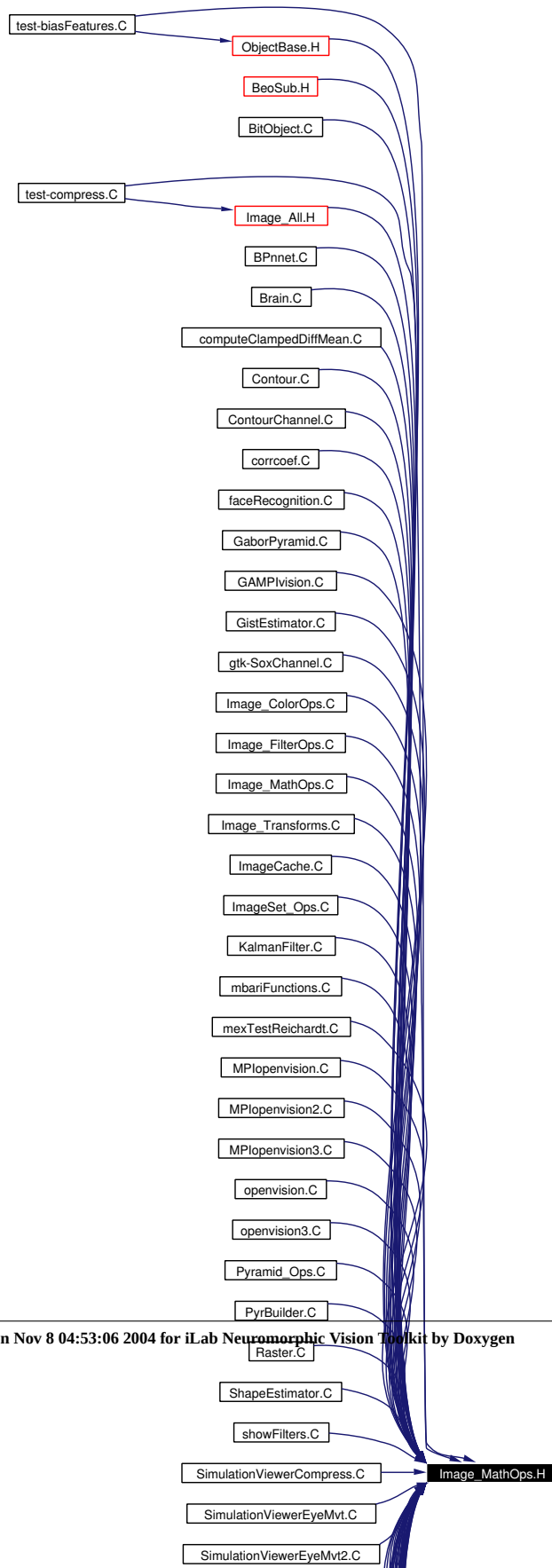
## 11.175 Image\_MathOps.H File Reference

```
#include <vector>
#include "saliency.H"
#include "Promotions.H"
#include "Point2D.H"
```

Include dependency graph for Image\_MathOps.H:



This graph shows which files directly or indirectly include this file:



## Functions

- `template<class T_or_RGB> bool operator== (const Image< T_or_RGB > &img1, const Image< T_or_RGB > &img2)`  
*Equality: true if the images are the same size and all pixels are equal.*
- `template<class T> double sum (const Image< T > &a)`  
*Returns the sum of all input pixels.*
- `template<class T> double mean (const Image< T > &a)`  
*Get the average of all input pixels.*
- `template<class T> Range< T > rangeOf (const Image< T > &img)`  
*Get the min/max range of the input pixels.*
- `template<class T> Image< T > remapRange (const Image< T > &img, const Range< T > &from, const Range< T > &to)`  
*Linearly remap pixels from the source to dest dynamic ranges.*
- `Image< byte > normalizeWithScale (const Image< float > &src)`  
*Converts a float image to a byte image, normalizing to [0.0-255.0].*
- `template<class T> Image< T > squared (const Image< T > &a)`  
*Square each element of the input image.*
- `template<class T> Image< typename promote_trait< T, float >::TP > toPower (const Image< T > &a, double pow)`  
*Raise each element of the input image to the given power.*
- `template<class T> Image< T > abs (const Image< T > &a)`  
*Take absolute value of every pixel.*
- `template<class T> Image< typename promote_trait< T, float >::TP > hmax-Activation (const Image< T > &a, const float sigma)`  
*Compute  $\exp(\text{pixel} / (2 * \sigma^2))$  for every pixel.*
- `template<class T> Image< T > absDiff (const Image< T > &b, const Image< T > &c)`  
*Returns  $\text{abs}(b - c)$ .*
- `template<class T> Image< PixRGB< T > > absDiff (const Image< PixRGB< T > > &b, const Image< PixRGB< T > > &c)`  
*Returns  $\text{abs}(b - c)$  for each color channel.*

- `template<class T> Image< T > clampedDiff (const Image< T > &b, const Image< T > &c)`  
Returns  $(b - c)$ , clamping to zero all negative values.
- `template<class T> Image< PixRGB< T > > clampedDiff (const Image< PixRGB< T > > &b, const Image< PixRGB< T > > &c)`  
Returns  $(b - c)$  for each color channel, clamping to zero all neg. values.
- `template<class T> Image< T > binaryReverse (const Image< T > &a, const T val)`  
Returns  $(val - a)$ .
- `template<class T> Image< T > average (const Image< T > &b, const Image< T > &c)`  
Returns  $(b + c) / 2$ , clean & prevents overflows.
- `template<class T> Image< T > averageWeighted (Image< T > &a, const Image< T > &b, T *aWeight)`  
Returns  $(a*w + b) / 2$ , the weighted average between images.
- `template<class T> Image< T > takeMax (const Image< T > &a, const Image< T > &b)`  
Returns  $\max(a, b)$ , element by element.
- `template<class T> Image< PixRGB< T > > takeMax (const Image< PixRGB< T > > &a, const Image< PixRGB< T > > &b)`  
Returns  $\max(a, b)$  for each color component separately.
- `template<class T> Image< T > takeMin (const Image< T > &a, const Image< T > &b)`  
Returns  $\min(a, b)$ , element by element.
- `template<class T_or_RGB> Image< T_or_RGB > quadEnergy (const Image< T_or_RGB > &img1, const Image< T_or_RGB > &img2)`  
Compute energy for quadrature pair ( $\sqrt{\text{img1}^2 + \text{img2}^2}$ ).
- `template<class T> float RMSerr (const Image< T > &arr1, const Image< T > &arr2)`  
RMS error between arr1 and arr2.
- `template<class T> Image< T > overlay (const Image< T > &top, const Image< T > &bottom, const float trans)`

Create transparency overlay of two images.

- `template<class T> float sumXY (const Image< T > &img, std::vector< float > &sumX, std::vector< float > &sumY)`  
*sum pixels along the X and Y axes (sumX, sumY) and in total (return value)*
- `template<class T> double distance (const Image< T > &img1, const Image< T > &img2)`  
*Returns Euclidean distance between two images.*
- `template<class T> double distance (const Image< T > &img1, const Image< T > &img2, const Image< float > &weight)`  
*Returns weighted Euclidean distance between two images.*
- `double corrcoef (const Image< float > &img1, const Image< float > &img2)`  
*Returns the correlation coefficient  $r^2$  of the two images.*
- `template<class T> double corrpatch (const Image< T > &img1, const Point2D &toleft1, const Dims &patchdims, const Image< T > &img2, const Point2D &toleft2, const double eps=1.0e-20)`  
*Returns the cross-correlation coefficient  $r$  between two patches.*
- `template<class T> double pSNR (const Image< T > &img1, const Image< T > &img2)`  
*Returns peak signal-to-noise ratio.*
- `template<class T> double pSNR (const Image< T > &img1, const Image< T > &img2, const Image< float > &weight)`  
*Returns weighted peak signal-to-noise ratio.*
- `template<class T> Image< typename promote\_trait< T, float >::TP > sqrt (const Image< T > &img)`  
*Take square root of all elements.*
- `template<class T> Image< typename promote\_trait< T, float >::TP > inverse (const Image< T > &img, const T eps)`  
*Take inverse of all elements > eps in abs, 0 otherwise.*
- `template<class T> Image< typename promote\_trait< T, float >::TP > exp (const Image< T > &img)`  
*Take exponential of all pixels.*
- `template<class T> Image< typename promote\_trait< T, float >::TP > negexp (const Image< T > &img)`



*Take negative exponential of all pixels, that is  $\exp(-x)$ .*

- `template<class T> bool getCentroidFirstLast (std::vector< T > vect, float &centroid, int &first, int &last)`

*determine the first and last non-zero values and the centroid*

- `template<class T> Point2D centroid (const Image< T > &img, Rectangle &boundingBox, float &cenX, float &cenY)`

*determine the centroid of all the points in the image*

- `template<class T> Point2D centroid (const Image< T > &img)`

*determine the center of mass of all the points in the image*

- `template<class T> Image< T > squash (const Image< T > &src, const T oldmin, const T newmin, const T oldmid, const T newmid, const T oldmax, const T newmax)`

*Apply a squashing function to the pixel values.*

- `template<class T> Image< T > squash (const Image< T > &src, const T newmin, const T oldmid, const T newmid, const T newmax)`

*Apply a squashing function to the pixel values.*

- `template<class T> Image< T > squash (const Image< T > &src, const T oldmid, const T newmid)`

*Apply a squashing function to the pixel values.*

- `template<class T, class T_or_RGB> Image< T_or_RGB > thresholdedMix (const Image< T > &mask, const T &thresh, const Image< T_or_RGB > &lower, const Image< T_or_RGB > &higher)`

*Mix between two images based on comparing mask values to a threshold.*

- `Image< float > matrixMult (const Image< float > &A, const Image< float > &B)`

*Matrix Multiplication:  $C = A * B$ .*

- `Image< float > matrixMult (const Image< float > &A, const Image< float > &B, unsigned int *beginAX, unsigned int *endAX, unsigned int *beginBY, unsigned int *endBY, unsigned int *beginAY, unsigned int *endAY)`

*Matrix Multiplication:  $C = A * B$ .*

- `Image< float > matrixMult (const Image< float > &A, const Image< float > &B, unsigned long *beginAX, unsigned long *endAX, unsigned long *beginBY, unsigned long *endBY, unsigned long *beginAY, unsigned long *endAY)`

*Matrix Multiplication:  $C = A * B$ .*

- `template<class T> std::vector< std::vector< T > > matrixMult (std::vector< std::vector< T > > *A, std::vector< std::vector< T > > *B, unsigned long *beginAX, unsigned long *endAX, unsigned long *beginBX, unsigned long *endBX, unsigned long *beginAY, unsigned long *endAY)`

*Matrix Multiplication:  $C = A * B$ .*

- `template<class T_or_RGB> Image< T_or_RGB > transpose (const Image< T_or_RGB > &M)`

*transpose matrix M*

- `Image< float > eye (const int size)`

*return the identity matrix of dimensions (size x size)*

- `template<class T_or_RGB> Image< T_or_RGB > scramble (const Image< T_or_RGB > &ima)`

*randomly scramble image pixels*

- `int32 findMontyMap (const Image< float > &ima, std::vector< Point2D > *coords, float bias)`

*take in an image and return it's statistically most relevent points*

- `int32 makeSparseMap (std::vector< Point2D > *coords, std::vector< Point2D * > *cmap, std::vector< bool > *keep, int inPoints, int outPoints)`

*Take in a vector and decimate it according how many points you want back.*

### 11.175.1 Detailed Description

Mathematical operations on [Image](#)

Definition in file [Image\\_MathOps.H](#).

### 11.175.2 Function Documentation

#### 11.175.2.1 `template<class T> Image<T> abs (const Image< T > &a)`

Take absolute value of every pixel.

Definition at line 216 of file [Image\\_MathOps.C](#).

References [Image](#)< T >::begin(), [Image](#)< T >::beginw(), [Image](#)< T >::end(), [Image](#)< T >::getDims(), and NO\_INIT.

Referenced by `Point< T, N >::abs()`, `SingleChannel::absSumCoeffs()`, `DirectFeedChannel::absSumCoeffs()`, `buildPyrGabor()`, `Image< T >::lowThreshAbs()`, `Point< T, N >::max()`, `CameraControl::moveCamXYFrame()`, `operator>>()`, and `GACHromosome::set_gene()`.

#### 11.175.2.2 `template<class T> Image< PixRGB<T> > absDiff (const Image< PixRGB< T > > & b, const Image< PixRGB< T > > & c)`

Returns `abs(b - c)` for each color channel.

Definition at line 265 of file `Image_MathOps.C`.

References `ASSERT`, `Image< T >::beginw()`, `Image< T >::endw()`, and `NO_INIT`.

Referenced by `ImageCacheAvg< T >::absDiffMean()`, `FlickerChannel::doInput()`, `TaskRelevanceMapKillStatic::evolve()`, and `main()`.

#### 11.175.2.3 `template<class T> Image<T> absDiff (const Image< T > & b, const Image< T > & c)`

Returns `abs(b - c)`.

Definition at line 243 of file `Image_MathOps.C`.

References `ASSERT`, `Image< T >::begin()`, `Image< T >::beginw()`, `Image< T >::endw()`, `Image< T >::getDims()`, `Image< T >::isSameSize()`, and `NO_INIT`.

#### 11.175.2.4 `template<class T> Image<T> average (const Image< T > & b, const Image< T > & c)`

Returns `(b + c) / 2`, clean & prevents overflows.

Definition at line 464 of file `Image_MathOps.C`.

References `ASSERT`, `Image< T >::begin()`, `Image< T >::beginw()`, `Image< T >::endw()`, `Image< T >::getDims()`, `Image< T >::isSameSize()`, and `NO_INIT`.

#### 11.175.2.5 `template<class T> Image<T> averageWeighted (Image< T > & a, const Image< T > & b, T * aWeight)`

Returns `(a*w + b) / 2`, the weighted average between images.

Definition at line 482 of file `Image_MathOps.C`.

References `ASSERT`, `Image< T >::begin()`, `Image< T >::beginw()`, `Image< T >::endw()`, and `Image< T >::isSameSize()`.

### 11.175.2.6 `template<class T> Image<T> binaryReverse (const Image< T > & a, const T val)`

Returns (val - a).

Definition at line 447 of file Image\_MathOps.C.

References Image< T >::begin(), Image< T >::beginw(), Image< T >::endw(), Image< T >::getDims(), and NO\_INIT.

Referenced by SimulationViewerEyeMvt2::attentionShift(), SimulationViewerEyeMvt2::eyeMovement(), SingleChannel::getRawCSmap(), ShapeEstimator::getSmoothMask(), SimulationViewerCompress::getTraj(), VisualBuffer::inhibit(), VisualBuffer::input(), TaskRelevanceMapKillN::input(), VisualCortex::learn(), and learningCoeff().

### 11.175.2.7 `template<class T> Point2D centroid (const Image< T > & img)`

determine the center of mass of all the points in the image

The centroid is rounded to integer pixel positions. When there are no values > 0 in the img, (-1,-1) is returned

Definition at line 904 of file Image\_MathOps.C.

Referenced by VisualBuffer::findMostInterestingTarget(), and main().

### 11.175.2.8 `template<class T> Point2D centroid (const Image< T > & img, Rectangle & boundingBox, float & cenX, float & cenY)`

determine the centroid of all the points in the image

The centroid is rounded to integer pixel positions. When there are no values > 0 in the img, (-1,-1) is returned

#### Parameters:

***boundingBox*** the bounding box of the object(s) contained in the image is returned

***cenX*** returns the float value of the centroid along the x axis

***cenY*** returns the float value of the centroid along the y axis

#### Returns:

the centroid rounded to integer coordinates

Definition at line 881 of file Image\_MathOps.C.

References getCentroidFirstLast(), Rectangle::modify(), and sumXY().

**11.175.2.9** `template<class T> Image< PixRGB<T> > clampedDiff (const Image< PixRGB< T > > & b, const Image< PixRGB< T > > & c)`

Returns (b - c) for each color channel, clamping to zero all neg. values.

Definition at line 414 of file Image\_MathOps.C.

References ASSERT, Image< T >::beginw(), Image< T >::endw(), and NO\_INIT.

**11.175.2.10** `template<class T> Image<T> clampedDiff (const Image< T > & b, const Image< T > & c)`

Returns (b - c), clamping to zero all negative values.

Definition at line 334 of file Image\_MathOps.C.

References ASSERT, Image< T >::begin(), Image< T >::beginw(), Image< T >::endw(), Image< T >::getDims(), Image< T >::isSameSize(), and NO\_INIT.

**11.175.2.11** `double corrcoeff (const Image< float > & img1, const Image< float > & img2)`

Returns the correlation coefficient  $r^2$  of the two images.

Definition at line 676 of file Image\_MathOps.C.

References ASSERT, Image< T >::begin(), Image< T >::end(), Image< T >::getSize(), Image< T >::isSameSize(), and SQR().

**11.175.2.12** `template<class T> double corrpitch (const Image< T > & img1, const Point2D & topleft1, const Dims & patchdims, const Image< T > & img2, const Point2D & topleft2, const double eps = 1.0e-20)`

Returns the cross-correlation coefficient r between two patches.

Note that the computations done here differ from what is done in corrcoeff, in that a signed correlation coefficient is returned (between -1 and 1), and we add a convention that if both patches have zero variance then the correlation is 1. If only one patch has zero variance then the correlation is zero. Current implementation requires that the patch fits within both image bounds, and assumes a small patch size so that it will not attempt to use a faster FFT-based implementation which can save time with larger patches.

#### Parameters:

*img1* first image

*topleft1* coordinates of top-left corner of patch in img1

*patchdims* dimensions of patch

*img2* second image

*toleft2* coordinates of top-left corner of patch in *img2*

*eps* epsilon used to determine whether patch variance is zero

Definition at line 724 of file Image\_MathOps.C.

References ASSERT, Image< T >::begin(), Image< T >::getWidth(), Dims::h(), Point2D::i, Point2D::j, Image< T >::rectangleOk(), sqrt(), and Dims::w().

Referenced by TcorrChannel::getOutput().

#### 11.175.2.13 **template<class T> double distance (const Image< T > & *img1*, const Image< T > & *img2*, const Image< float > & *weight*)**

Returns weighted Euclidean distance between two images.

Returns  $\sqrt{\text{sum\_over\_pixels}(\text{weight}[i] * (\text{img1}[i] - \text{img2}[i])^2)}$

Definition at line 658 of file Image\_MathOps.C.

References ASSERT, Image< T >::begin(), Image< T >::end(), Image< T >::is-SameSize(), and sqrt().

Referenced by ImageSpring< T >::computeStiff(), VisualCortex::getJetDistMap(), ImageSpring< T >::getStatsDist(), pSNR(), and pSNRcolor().

#### 11.175.2.14 **template<class T> double distance (const Image< T > & *img1*, const Image< T > & *img2*)**

Returns Euclidean distance between two images.

Returns  $\sqrt{\text{sum\_over\_pixels}((\text{img1}[i] - \text{img2}[i])^2)}$

Definition at line 643 of file Image\_MathOps.C.

References ASSERT, Image< T >::begin(), Image< T >::end(), Image< T >::is-SameSize(), and sqrt().

#### 11.175.2.15 **template<class T> Image<typename promote\_trait<T,float>::TP> exp (const Image< T > & *img*)**

Take exponential of all pixels.

Definition at line 817 of file Image\_MathOps.C.

References Image< T >::begin(), Image< T >::beginw(), Image< T >::end(), Image< T >::getDims(), and NO\_INIT.

Referenced by SingleChannel::centerSurround(), MonkeySaccadeController::computeWhenNewDecision(), VisualObject::createKeys(), distDegrade(), dogFilter(), fixationMask(), gabor2(), gaborFilter(), gaborFilter2(), gaborFilter3(), gaussian(), gaussian2D(), hmaxActivation(), hueDistance(), longRangeExcFilter(), negexp(), MonkeyEyeHeadController::saccade(), CarControl::toSpeedSigmoid(), and SaliencyMap::w().

#### 11.175.2.16 **Image<float> eye (const int size)**

return the identity matrix of dimensions (size x size)

Definition at line 1150 of file Image\_MathOps.C.

References Image< T >::beginw(), and ZEROS.

Referenced by MonkeySaccadeController::computeWhenNewDecision(), RetinaStd::doGetOutput(), KalmanFilter::init(), and MonkeySaccadeController::MonkeySaccadeController().

#### 11.175.2.17 **int32 findMontyMap (const Image< float > & ima, std::vector< Point2D > \* coords, float bias)**

take in an image and return it's statistically most relevent points

Input an image to find a monte carlo like map. This will in essence cut out about half the pixels in a simple quick fashion by applying the formula: select if -> pow(pixel-Value,bias) > minVal+(maxVal\*rand()/RAND\_MAX+1.0) The bias is used to scew the distribution using an exponent if no bias is given, this method will work more effciently by skipping the exponent computation over each pixel.

##### Parameters:

**ima** This is the input probability map

**coords** this is a pointer to a vector which will hold selected coords

**bais** this scews the probability map by pow(pixVal,bias)

Definition at line 1183 of file Image\_MathOps.C.

References ASSERT, Image< PixRGB< byte > >::begin(), Image< PixRGB< byte > >::getSize(), and Image< PixRGB< byte > >::getWidth().

#### 11.175.2.18 **template<class T> bool getCentroidFirstLast (std::vector< T > vect, float & centroid, int & first, int & last)**

determine the first and last non-zero values and the centroid

##### Parameters:

**vect** a vector of numbers

***centroid*** returns the centroid of vect

***first*** returns the position of the first non-zero value in vect

***last*** returns the position of the last non-zero value in vect

**Returns:**

true if we actually found all these values, false if vect only contains zeros

Definition at line 841 of file Image\_MathOps.C.

References centroid(), and uint.

Referenced by centroid(), and BitObject::reset().

**11.175.2.19** `template<class T> Image<typename promote_  
trait<T,float>::TP> hmaxActivation (const Image< T > & a,  
const float sigma)`

Compute  $\exp(\text{pixel} / (2 * \sigma^2))$  for every pixel.

Definition at line 226 of file Image\_MathOps.C.

References Image< T >::begin(), Image< T >::beginw(), exp(), Image< T >::getDims(), Image< T >::getSize(), and NO\_INIT.

Referenced by Hmax::origGetC2().

**11.175.2.20** `template<class T> Image<typename promote_  
trait<T,float>::TP> inverse (const Image< T > & img, const T  
eps)`

Take inverse of all elements > eps in abs, 0 otherwise.

Definition at line 807 of file Image\_MathOps.C.

References Image< T >::begin(), Image< T >::beginw(), Image< T >::end(), Image< T >::getDims(), and NO\_INIT.

Referenced by ImageSpring< T >::getStats().

**11.175.2.21** `int32 makeSparseMap (std::vector< Point2D > * coords,  
std::vector< Point2D * > * cmap, std::vector< bool > * keep, int  
inPoints, int outPoints)`

Take in a vector and decimate it according how many points you want back.

In essence this will decimate a list of coordinates attempting to create a list of the size in outPoints. Thus, if you have list of 23890 points and want a list of 300 points, this will take every nth point and create a list of about this size.



**Parameters:**

- coords*** This is the list of input coordinates
- cmap*** This is the final list of coordinates after decimation
- inPoints*** this is the input list size
- outPoints*** this is how many points you would like out

Definition at line 1248 of file Image\_MathOps.C.

References ASSERT.

```
11.175.2.22  template<class T> std::vector<std::vector<T> > matrixMult
              (std::vector< std::vector< T > > * A, std::vector< std::vector< T
              > > * B, unsigned long * beginAX, unsigned long * endAX,
              unsigned long * beginBX, unsigned long * endBX, unsigned long *
              beginAY, unsigned long * endAY)
```

Matrix Multiplication:  $C = A * B$ .

The images A and B are interpreted as matrices. For multiplying a (m x n) with a (n x p) matrix, this is the generic matrix multiplication with complexity (m \* n \* p). This version will allow you to multiply values in a matrix that lie between index endA/B and beginA/B. The destination matrix will be the same size as: (endAX - beginAX) x (endAX - beginAX) per usual the width of A (indexed between beginAX and endAX) and height of B MUST be equal. If not then perhaps use transpose... This method is for large matrices

**Parameters:**

- A*** input matrix 1
- B*** input matrix 2
- beginAX*** starting (width) index in matrix A
- endAX*** ending (width) index in matrix A
- beginBY*** starting (height) index in matrix B
- endBY*** ending (height) index in matrix B
- beginAY*** starting (height) index in matrix A
- endAY*** ending (height) index in matrix A

Definition at line 1088 of file Image\_MathOps.C.

References ASSERT, B, and ZEROS.

**11.175.2.23** `Image<float> matrixMult (const Image< float > & A, const Image< float > & B, unsigned long * beginAX, unsigned long * endAX, unsigned long * beginBY, unsigned long * endBY, unsigned long * beginAY, unsigned long * endAY)`

Matrix Multiplication:  $C = A * B$ .

The images A and B are interpreted as matrices. For multiplying a (m x n) with a (n x p) matrix, this is the generic matrix multiplication with complexity (m \* n \* p). This version will allow you to multiply values in a matrix that lie between index endA/B and beginA/B. The destination matrix will be the same size as: (endAX - beginAX) x (endAX - beginAX) per usual the width of A (indexed between beginAX and endAX) and height of B MUST be equal. If not then perhaps use transpose... This method is for large matrices

**Parameters:**

*A* input matrix 1  
*B* input matrix 2  
*beginAX* starting (width) index in matrix A  
*endAX* ending (width) index in matrix A  
*beginBY* starting (height) index in matrix B  
*endBY* ending (height) index in matrix B  
*beginAY* starting (height) index in matrix A  
*endAY* ending (height) index in matrix A

Definition at line 1058 of file Image\_MathOps.C.

References ASSERT, B, Image< T >::beginw(), Image< T >::getVal(), Image< T >::initialized(), and ZEROS.

**11.175.2.24** `Image<float> matrixMult (const Image< float > & A, const Image< float > & B, unsigned int * beginAX, unsigned int * endAX, unsigned int * beginBY, unsigned int * endBY, unsigned int * beginAY, unsigned int * endAY)`

Matrix Multiplication:  $C = A * B$ .

The images A and B are interpreted as matrices. For multiplying a (m x n) with a (n x p) matrix, this is the generic matrix multiplication with complexity (m \* n \* p). This version will allow you to multiply values in a matrix that lie between index endA/B and beginA/B. The destination matrix will be the same size as: (endAX - beginAX) x (endAX - beginAX) per usual the width of A (indexed between beginAX and endAX) and height of B MUST be equal. If not then perhaps use transpose...

**Parameters:***A* input matrix 1*B* input matrix 2*beginAX* starting (width) index in matrix A*endAX* ending (width) index in matrix A*beginBY* starting (height) index in matrix B*endBY* ending (height) index in matrix B*beginAY* starting (height) index in matrix A*endAY* ending (height) index in matrix A

Definition at line 1030 of file Image\_MathOps.C.

References ASSERT, B, Image< T >::beginw(), Image< T >::getVal(), Image< T >::initialized(), and ZEROS.

#### 11.175.2.25 **Image<float> matrixMult (const Image< float > & A, const Image< float > & B)**

Matrix Multiplication:  $C = A * B$ .

The images A and B are interpreted as matrices. For multiplying a (m x n) with a (n x p) matrix, this is the generic matrix multiplication with complexity (m \* n \* p).

Definition at line 1010 of file Image\_MathOps.C.

References ASSERT, B, Image< T >::beginw(), Image< T >::getHeight(), Image< T >::getVal(), Image< T >::getWidth(), Image< T >::initialized(), and ZEROS.

#### 11.175.2.26 **template<class T> double mean (const Image< T > & a)**

Get the average of all input pixels.

Definition at line 107 of file Image\_MathOps.C.

References ASSERT, Image< T >::getSize(), and sum().

#### 11.175.2.27 **template<class T> Image<typename promote\_ - trait<T,float>::TP> negexp (const Image< T > & img)**

Take negative exponential of all pixels, that is  $\exp(-x)$ .

Definition at line 829 of file Image\_MathOps.C.

References Image< T >::begin(), Image< T >::beginw(), Image< T >::end(), exp(), Image< T >::getDims(), and NO\_INIT.

**11.175.2.28 `Image<byte> normalizeWithScale (const Image<float> & src)`**

Converts a float image to a byte image, normalizing to [0.0-255.0].

Two lines of text are appended to the output indicating the original min and max values of the float image.

Definition at line 142 of file Image\_MathOps.C.

References Image< T >::getHeight(), Image< T >::getWidth(), Range< T >::max(), Range< T >::min(), Image< T >::normalize(), Image< T >::paste(), rangeOf(), Image< T >::writeText(), and ZEROS.

Referenced by Raster::WriteFloatNorm().

**11.175.2.29 `template<class T_or_RGB> bool operator== (const Image< T_or_RGB> & img1, const Image< T_or_RGB> & img2)`**

Equality: true if the images are the same size and all pixels are equal.

**11.175.2.30 `template<class T> Image<T> overlay (const Image< T> & top, const Image< T> & bottom, const float trans)`**

Create transparency overlay of two images.

by laying top over bottom with transparency of top set to trans. A new image will be returned (use 0 to 100 for percent transparency). NOTE: top and bottom must be the same size! this function only works for grey scale images

Definition at line 587 of file Image\_MathOps.C.

References ASSERT, Image< T >::begin(), Image< T >::beginw(), Image< T >::endw(), Image< T >::getDims(), Image< T >::isSameSize(), and NO\_INIT.

**11.175.2.31 `template<class T> double pSNR (const Image< T> & img1, const Image< T> & img2, const Image<float> & weight)`**

Returns weighted peak signal-to-noise ratio.

This is like the other `pSNR()` except that it uses the weighted version of `distance()` internally

Definition at line 785 of file Image\_MathOps.C.

References distance(), and Image< T >::getSize().

**11.175.2.32** `template<class T> double pSNR (const Image< T > & img1, const Image< T > & img2)`

Returns peak signal-to-noise ratio.

pSNR is computed as  $10 \cdot \log_{10}(255^2 / \sigma^2)$  where  $\sigma^2 = 1/N \cdot \sum (x1_i - x2_i)^2$  and is a measure of distortion between two images

Definition at line 776 of file Image\_MathOps.C.

References `distance()`, and `Image< T >::getSize()`.

**11.175.2.33** `template<class T_or_RGB> Image<T_or_RGB> quadEnergy (const Image< T_or_RGB > & img1, const Image< T_or_RGB > & img2)`

Compute energy for quadrature pair ( $\sqrt{img1^2 + img2^2}$ ).

**11.175.2.34** `template<class T> Range<T> rangeOf (const Image< T > & img)`

Get the min/max range of the input pixels.

This provides similar functionality to `Image<T>::getMinMax()`.

Definition at line 115 of file Image\_MathOps.C.

References `Image< T >::begin()`, and `Image< T >::end()`.

**11.175.2.35** `template<class T> Image<T> remapRange (const Image< T > & img, const Range< T > & from, const Range< T > & to)`

Linearly remap pixels from the source to dest dynamic ranges.

This is similar to, but more general than, `Image<T>::normalize()`. In effect, `Image<T>::normalize()` always assumes that the source range is the image's own min/max range. In contrast, `remapRange()` allows one to find the global min/max across a range of images, and then use that as the source range while normalizing each the images using that range.

Definition at line 122 of file Image\_MathOps.C.

References `Image< T >::begin()`, `Image< T >::beginw()`, `Image< T >::endw()`, `Image< T >::getDims()`, `Range< T >::min()`, `NO_INIT`, and `Range< T >::range()`.

**11.175.2.36** `template<class T> float RMSerr (const Image< T > & arr1, const Image< T > & arr2)`

RMS error between *arr1* and *arr2*.

Definition at line 570 of file Image\_MathOps.C.

References ASSERT, Image< T >::begin(), Image< T >::end(), Image< T >::getSize(), Image< T >::initialized(), Image< T >::isSameSize(), and sqrt().

**11.175.2.37** `template<class T_or_RGB> Image<T_or_RGB> scramble (const Image< T_or_RGB > & ima)`

randomly scramble image pixels

**11.175.2.38** `template<class T> Image<typename promote_trait<T,float>::TP> sqrt (const Image< T > & img)`

Take square root of all elements.

Definition at line 795 of file Image\_MathOps.C.

References Image< T >::begin(), Image< T >::beginw(), Image< T >::end(), Image< T >::getDims(), NO\_INIT, and sqrt().

**11.175.2.39** `template<class T> Image<T> squared (const Image< T > & a)`

Square each element of the input image.

Definition at line 175 of file Image\_MathOps.C.

References Image< T >::begin(), Image< T >::beginw(), Image< T >::end(), Image< T >::getDims(), and NO\_INIT.

Referenced by dogFilterHmax(), DisparityChannel::doInput(), and Hmax::init().

**11.175.2.40** `template<class T> Image<T> squash (const Image< T > & src, const T oldmid, const T newmid)`

Apply a squashing function to the pixel values.

In this specialization of the general form of [squash\(\)](#), the old min/max are computed using getMinMax(), and the new min/max will be kept equal to the old ones, so that there is no overall change in range (unless your mid-point is outside that range).

Definition at line 979 of file Image\_MathOps.C.

References Image< T >::getMinMax().

Referenced by VisualBuffer::input(), and squash().

**11.175.2.41** `template<class T> Image<T> squash (const Image< T > & src,  
const T newmin, const T oldmid, const T newmid, const T newmax)`

Apply a squashing function to the pixel values.

In this specialization of the general form of `squash()`, the old min/max are computed using `getMinMax()`.

Definition at line 967 of file `Image_MathOps.C`.

References `Image< T >::getMinMax()`, and `squash()`.

**11.175.2.42** `template<class T> Image<T> squash (const Image< T > & src,  
const T oldmin, const T newmin, const T oldmid, const T newmid,  
const T oldmax, const T newmax)`

Apply a squashing function to the pixel values.

The squashing function is a 4th-order polynomial with sigmoidal shape. Denoting by  $x$  a pixel value in the original image and by  $y=f(x)$  the transformed pixel value that will be put in the result image, the constraints on the polynomial are: 1)  $f(\text{oldmin})=\text{newmin}$ , 2)  $f(\text{oldmax})=\text{newmax}$ , 3)  $f(\text{oldmid})=\text{newmid}$ , 4)  $f'(\text{oldmin})=0$ , and 5)  $f'(\text{oldmax})=0$ . So, typically, the polynomial will have horizontal slope at both ends of the input range, and will remap the old min/max to the new min/max while also remapping a user-chosen mid-point `oldmid` to `newmid`. Playing with this mid-point allows the user to give more emphasis to higher or lower values while remapping. This is the full form, but two simplified forms are also available that will compute some of the arguments here from the input image. Because the main use of this function is when `oldmin` is the true minimum value of the input image, and `oldmax` its true maximum, here we will not waste time checking for input values outside this range, and their remapped values are unclear as the polynomial is unconstrained outside the input range. Internal remapping is done using floats, and results are converted back (with possible clamping) to the same type as the original image, so that this function may be efficiently used to do contrast adjustments in `Image<byte>` data.

Definition at line 913 of file `Image_MathOps.C`.

References `Image< PixRGB< byte > >::begin()`, `Image< T >::beginw()`, `Image< PixRGB< byte > >::end()`, `Image< PixRGB< byte > >::getDims()`, and `NO_INIT`.

**11.175.2.43** `template<class T> double sum (const Image< T > & a)`

Returns the sum of all input pixels.

Definition at line 66 of file `Image_MathOps.C`.

References `Image< T >::begin()`, and `Image< T >::end()`.

**11.175.2.44** `template<class T> float sumXY (const Image< T > & img,  
std::vector< float > & sumX, std::vector< float > & sumY)`

sum pixels along the X and Y axes (sumX, sumY) and in total (return value)

**Parameters:**

*img* is a w x h size image

*sumX* will return a vector of length w with the sum over the rows of img

*sumY* will return a vector of length h with the sum over the columns

Definition at line 614 of file Image\_MathOps.C.

References Image< T >::begin(), Image< T >::getHeight(), and Image< T >::getWidth().

Referenced by centroid(), and BitObject::reset().

**11.175.2.45** `template<class T> Image< PixRGB<T> > takeMax (const  
Image< PixRGB< T > > & a, const Image< PixRGB< T > > & b)`

Returns max(a, b) for each color component separately.

Definition at line 515 of file Image\_MathOps.C.

References getComponents(), and makeRGB().

Referenced by SimulationViewerStd::attentionShift(), VisualCortexEyeMvt::evolve(), extractBitObjects(), SimulationViewerStd::eyeMovement(), ImageCacheMinMax< T >::getMax(), VisualCortexEyeMvt::getOutput(), ShapeEstimator::getSmoothMask(), SimulationViewerStd::headMovement(), VisualBuffer::input(), main(), maxRGB(), and showAllObjects().

**11.175.2.46** `template<class T> Image<T> takeMax (const Image< T > & a,  
const Image< T > & b)`

Returns max(a, b), element by element.

Definition at line 494 of file Image\_MathOps.C.

References ASSERT, Image< T >::begin(), Image< T >::beginw(), Image< T >::endw(), Image< T >::getDims(), Image< T >::isSameSize(), and NO\_INIT.

**11.175.2.47** `template<class T> Image<T> takeMin (const Image< T > & a,  
const Image< T > & b)`

Returns min(a, b), element by element.



Definition at line 527 of file Image\_MathOps.C.

References ASSERT, Image< T >::begin(), Image< T >::beginw(), Image< T >::endw(), Image< T >::getDims(), Image< T >::isSameSize(), and NO\_INIT.

Referenced by BitObject::doesIntersect(), ImageCacheMinMax< T >::getMin(), and SimulationViewerCompress::getTraj().

**11.175.2.48** `template<class T, class T_or_RGB> Image<T_or_RGB>  
thresholdedMix (const Image< T > & mask, const T & thresh,  
const Image< T_or_RGB > & lower, const Image< T_or_RGB >  
& higher)`

Mix between two images based on comparing mask values to a threshold.

On every pixel, if mask >= thresh, the returned value is taken from higher, otherwise it is taken from lower. All three of lower, highr and mask must have same dims.

Referenced by SimulationViewerCompress::getTraj().

**11.175.2.49** `template<class T> Image<typename promote_  
trait<T,float>::TP> toPower (const Image< T > & a, double  
pow)`

Raise each element of the input image to the given power.

Definition at line 207 of file Image\_MathOps.C.

References Image< T >::begin(), Image< T >::beginw(), Image< T >::end(), Image< T >::getDims(), and NO\_INIT.

Referenced by SoxChannel::getNonlinearResponse(), and SurpriseMap< T >::surprise().

**11.175.2.50** `template<class T_or_RGB> Image<T_or_RGB> transpose (const  
Image< T_or_RGB > & M)`

transpose matrix M

Definition at line 1128 of file Image\_MathOps.C.

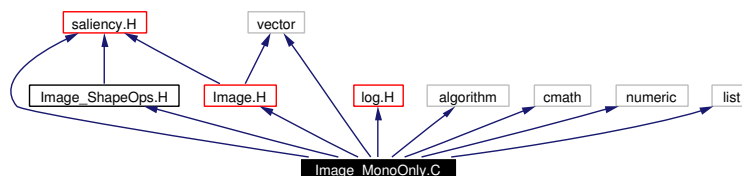
References Image< T >::begin(), Image< T >::beginw(), Image< T >::getHeight(), Image< T >::getWidth(), and NO\_INIT.

Referenced by KalmanFilter::init().

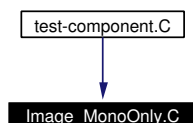
## 11.176 Image\_MonoOnly.C File Reference

```
#include "saliency.H"
#include "Image.H"
#include "Image_ShapeOps.H"
#include "log.H"
#include <algorithm>
#include <cmath>
#include <numeric>
#include <list>
#include <vector>
```

Include dependency graph for Image\_MonoOnly.C:



This graph shows which files directly or indirectly include this file:



### Defines

- #define **BGNOISELEVEL** 0.00001

### 11.176.1 Detailed Description

Operations for monochrome [Image](#)

Definition in file [Image\\_MonoOnly.C](#).

## 11.177 Image\_Omni.C File Reference

```
#include "Image_Omni.H"
#include "Image.H"
#include <cmath>
#include <iostream>
#include "inst/Image_Omni.I"
```

Include dependency graph for Image\_Omni.C:

### Functions

- template<class T> void [lowPassPixel](#) ([Image](#)< T > &src, const int x, const int y, const int radius)

*Perform a low pass filter about a specific pixel.*

- template<class T> [Image](#)< [PixRGB](#)< T > > [omniCorrectGen](#) (const [Image](#)< [PixRGB](#)< T > > &src, const int Rx, const int Ry, const int Xc, const int Yc, const int RA)

*First step of omni directional correction.*

- template<class T> [Image](#)< [PixRGB](#)< T > > [omniCorrectSp](#) (const [Image](#)< [PixRGB](#)< T > > &src, const float r, const float hh, const float kk, const int, const int, const int Xc, const int Yc)

*Second step of omni-directional correcton.*

- template<class T> [Image](#)< [PixRGB](#)< T > > [omniDenebulize](#) (const [Image](#)< [PixRGB](#)< T > > &src, const int radius)

*Denebulize a color [Image](#) using targeted low pass filtering.*

- template<class T> [Image](#)< T > [omniDenebulize](#) (const [Image](#)< T > &src, const int radius)

*Denebulize a byte [Image](#) using targeted low pass filtering.*

### 11.177.1 Detailed Description

Operations for omnidirectional correction

Definition in file [Image\\_Omni.C](#).

## 11.177.2 Function Documentation

**11.177.2.1** `template<class T> void lowPassPixel (Image< T > & src, const int x, const int y, const int radius)`

Perform a low pass filter about a specific pixel.

**Parameters:**

- x* The x coord of the pixel
- y* The y coord of the pixel
- radius* The radius of the filter

Definition at line 50 of file Image\_Omni.C.

References Image< T >::getHeight(), Image< T >::getVal(), Image< T >::getWidth(), and Image< T >::setVal().

Referenced by omniDenebulize().

**11.177.2.2** `template<class T> Image<PixRGB<T> > omniCorrectGen (const Image< PixRGB< T > > & src, const int Rx, const int Ry, const int Xc, const int Yc, const int RA = 0)`

First step of omni directional correction.

**Parameters:**

- image* The image your are going to process
- Rx* The Radius in X - May be equal to Ry
- Ry* The Radius in Y - May be equal to Rx
- Xc* The center pixel x
- Yc* The center pixel y
- RA* Manual radius adjustment, this is used only for distortion adjustment.

**Note:**

omniCorrectSp must be used next for full correction unless image is a true pinhole image.

Definition at line 69 of file Image\_Omni.C.

References Image< PixRGB< T > >::getDims(), Dims::h(), Point2D::i, Point2D::j, NO\_INIT, Image< T >::setVal(), sqrt(), and Dims::w().

Referenced by Omni< T >::run().

**11.177.2.3** `template<class T> Image<PixRGB<T>> omniCorrectSp (const Image< PixRGB< T >> & src, const float r, const float hh, const float kk, const int Rx, const int Ry, const int Xc, const int Yc)`

Second step of omni-directional correcton.

Values for r,h,k must be obtained via measurement and derived via circular regression (see Mundhenk et al SPIE 2000) Note: run omniCorrectGen first.

**Parameters:**

- image* The image your are going to process
- r* The radius calculated by circular regression
- hh* The Circle center coord h (i.e. x)
- kk* The Circle center coord k (i.e. y)
- Rx* The Radius in X - May be equal to Ry
- Ry* The Radius in Y - May be equal to Rx
- Xc* The center pixel x
- Yc* The center pixel y

Definition at line 153 of file Image\_Omni.C.

References NO\_INIT, Image< T >::setVal(), and sqrt().

**11.177.2.4** `template<class T> Image<T> omniDenebulize (const Image< T > & image, const int radius)`

Denebulize a byte Image using targeted low pass filtering.

**Parameters:**

- image* a standard Image
- radius* The size of the filter

Definition at line 211 of file Image\_Omni.C.

References Image< T >::getHeight(), Image< T >::getVal(), Image< T >::getWidth(), and lowPassPixel().

Referenced by Omni< T >::run().

**11.177.2.5** `template<class T> Image<PixRGB<T>> omniDenebulize (const Image< PixRGB< T >> & src, const int radius)`

Denebulize a color Image using targeted low pass filtering.

**Parameters:**

*image* a standard [Image](#)

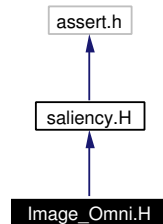
*radius* The size of the filter

Definition at line 190 of file Image\_Omni.C.

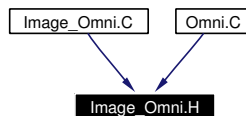
## 11.178 Image\_Omni.H File Reference

```
#include "saliency.H"
```

Include dependency graph for Image\_Omni.H:



This graph shows which files directly or indirectly include this file:



### Functions

- `template<class T> void lowPassPixel (Image< T > &src, const int x, const int y, const int radius)`  
*Perform a low pass filter about a specific pixel.*
- `template<class T> Image< PixRGB< T > > omniCorrectGen (const Image< PixRGB< T > > &src, const int Rx, const int Ry, const int Xc, const int Yc, const int RA=0)`  
*First step of omni directional correction.*
- `template<class T> Image< PixRGB< T > > omniCorrectSp (const Image< PixRGB< T > > &src, const float r, const float hh, const float kk, const int Rx, const int Ry, const int Xc, const int Yc)`  
*Second step of omni-directional correcton.*
- `template<class T> Image< PixRGB< T > > omniDenebulize (const Image< PixRGB< T > > &src, const int radius)`  
*Denebulize a color Image using targeted low pass filtering.*

- `template<class T> Image< T > omniDenebulize (const Image< T > &image, const int radius)`

*Denebulize a byte [Image](#) using targeted low pass filtering.*

### 11.178.1 Detailed Description

Operations for omnidirectional correction

Definition in file [Image\\_Omni.H](#).

### 11.178.2 Function Documentation

#### 11.178.2.1 `template<class T> void lowPassPixel (Image< T > & src, const int x, const int y, const int radius)`

Perform a low pass filter about a specific pixel.

**Parameters:**

- x* The x coord of the pixel
- y* The y coord of the pixel
- radius* The radius of the filter

Definition at line 50 of file [Image\\_Omni.C](#).

References [Image](#)< T >::getHeight(), [Image](#)< T >::getVal(), [Image](#)< T >::getWidth(), and [Image](#)< T >::setVal().

Referenced by [omniDenebulize\(\)](#).

#### 11.178.2.2 `template<class T> Image<PixRGB<T> > omniCorrectGen (const Image< PixRGB< T > > & src, const int Rx, const int Ry, const int Xc, const int Yc, const int RA = 0)`

First step of omni directional correction.

**Parameters:**

- image* The image your are going to process
- Rx* The Radius in X - May be equal to Ry
- Ry* The Radius in Y - May be equal to Rx
- Xc* The center pixel x



**Yc** The center pixel y

**RA** Manual radius adjustment, this is used only for distortion adjustment.

**Note:**

omniCorrectSp must be used next for full correction unless image is a true pinhole image.

Definition at line 69 of file Image\_Omni.C.

References Image< PixRGB< T > >::getDims(), Dims::h(), Point2D::i, Point2D::j, NO\_INIT, Image< T >::setVal(), sqrt(), and Dims::w().

Referenced by Omni< T >::run().

**11.178.2.3** `template<class T> Image<PixRGB<T> > omniCorrectSp (const Image< PixRGB< T > > & src, const float r, const float hh, const float kk, const int Rx, const int Ry, const int Xc, const int Yc)`

Second step of omni-directional correcton.

Values for r,h,k must be obtained via measurement and derived via circular regression (see Mundhenk et al SPIE 2000) Note: run omniCorrectGen first.

**Parameters:**

**image** The image your are going to process  
**r** The radius calculated by circular regression  
**hh** The Circle center coord h (i.e. x)  
**kk** The Circle center coord k (i.e. y)  
**Rx** The Radius in X - May be equal to Ry  
**Ry** The Radius in Y - May be equal to Rx  
**Xc** The center pixel x  
**Yc** The center pixel y

Definition at line 153 of file Image\_Omni.C.

References NO\_INIT, Image< T >::setVal(), and sqrt().

**11.178.2.4** `template<class T> Image<T> omniDenebulize (const Image< T > & image, const int radius)`

Denebulize a byte Image using targeted low pass filtering.

**Parameters:**

**image** a standard Image

*radius* The size of the filter

Definition at line 211 of file Image\_Omni.C.

References Image< T >::getHeight(), Image< T >::getVal(), Image< T >::getWidth(), and lowPassPixel().

Referenced by Omni< T >::run().

**11.178.2.5** `template<class T> Image<PixRGB<T> > omniDenebulize (const Image< PixRGB< T > > & src, const int radius)`

Denebulize a color Image using targeted low pass filtering.

**Parameters:**

*image* a standard Image

*radius* The size of the filter

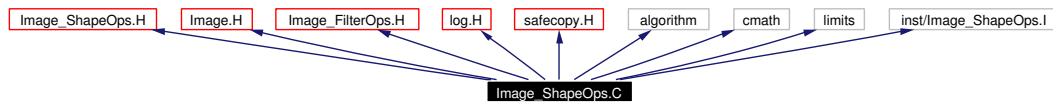
Definition at line 190 of file Image\_Omni.C.

References PixRGB< float >::blue(), Image< T >::getHeight(), Image< T >::getVal(), Image< T >::getWidth(), PixRGB< float >::green(), lowPassPixel(), and PixRGB< float >::red().

## 11.179 Image\_ShapeOps.C File Reference

```
#include "Image_ShapeOps.H"
#include "Image.H"
#include "Image_FilterOps.H"
#include "log.H"
#include "safecopy.H"
#include <algorithm>
#include <cmath>
#include <limits>
#include "inst/Image_ShapeOps.I"
```

Include dependency graph for Image\_ShapeOps.C:



## Functions

- `template<class T> Image< T > quickLocalAvg (const Image< T > &array, const int scale)`
- `template<class T> Image< T > quickLocalMax (const Image< T > &array, const int scale)`
- `template<class T> Image< T > quickLocalMin (const Image< T > &array, const int scale)`
- `template<class T> Image< T > quickInterpolate (const Image< T > &src, const int sfactor)`
- `template<class T> Image< T > rescale (const Image< T > &src, const Dims &dims)`
- `template<class T> Image< T > rescale (const Image< T > &src, const int new_w, const int new_h)`
- `template<class T> Image< T > downSize (const Image< T > &src, const Dims &dims)`

*Downsize using alternating lowpass/decimate operations.*

- `template<class T> Image< T > downSize (const Image< T > &src, const int new_w, const int new_h)`

*Downsize using alternating lowpass/decimate operations.*

- `Image< float > downSizeClean` (const `Image< float > &src`, const `Dims &new_dims`)

*Like `downSize()`, except image proportions don't have to match exactly.*

- `template<class T> Image< T > concatX` (const `Image< T > &left`, const `Image< T > &right`)
- `template<class T> Image< T > concatY` (const `Image< T > &top`, const `Image< T > &bottom`)
- `template<class T> Image< T > concatArray` (const `Image< T > arr[ ]`, const int nbarr, const int Nx, const int destX, const int destY)
- `template<class T> Image< T > decXY` (const `Image< T > &src`, const int factor)
- `template<class T> Image< T > decX` (const `Image< T > &src`, const int factor)
- `template<class T> Image< T > decY` (const `Image< T > &src`, const int factor)
- `template<class T> Image< T > intXY` (const `Image< T > &src`, const bool dupli)
- `template<class T> Image< T > intX` (const `Image< T > &src`, const bool dupli)
- `template<class T> Image< T > intY` (const `Image< T > &src`, const bool dupli)
- `template<class T> Image< T > zoomXY` (const `Image< T > &src`, int xzoom, int yzoom)
- `template<class T> Image< T > crop` (const `Image< T > &src`, const `Point2D &pt`, const `Dims &dims`, const bool zerofill)
- `template<class T> Image< T > crop` (const `Image< T > &src`, const `Rectangle &rect`, const bool zerofill)
- `template<class T> Image< T > shift` (const `Image< T > &srcImg`, const int dx, const int dy)
- `template<class T> Image< T > shiftImage` (const `Image< T > &srcImg`, const float dx, const float dy)
- `template<class T> Image< T > shiftClean` (const `Image< T > &srcImg`, const int dx, const int dy, const T bgval)
- `template<class T> Image< T > rotate` (const `Image< T > &srcImg`, const int x, const int y, const float ang)
- `template<class T> Image< T > combineImagesVertically` (const `Image< T > &topImg`, const `Image< T > &bottomImg`)

### 11.179.1 Detailed Description

Shape operations on `Image`

Definition in file `Image_ShapeOps.C`.

## 11.179.2 Function Documentation

### 11.179.2.1 `template<class T> Image<T> downSize (const Image< T > & src, const int new_w, const int new_h)`

Downsize using alternating lowpass/decimate operations.

Definition at line 244 of file Image\_ShapeOps.C.

References `ASSERT`, `decX()`, `decY()`, `Image< T >::getHeight()`, `Image< T >::getWidth()`, `lowPassX()`, and `lowPassY()`.

Referenced by `Image< T >::addDownsized()`, `computeCMAP()`, `InferoTemporal::computeVisualObject()`, `downSize()`, `SimulationViewerCompress::getTraj()`, `maxNormalizeFancy()`, and `SingleChannel::postProcessMap()`.

### 11.179.2.2 `template<class T> Image<T> downSize (const Image< T > & src, const Dims & dims)`

Downsize using alternating lowpass/decimate operations.

Definition at line 237 of file Image\_ShapeOps.C.

### 11.179.2.3 `Image<float> downSizeClean (const Image< float > & src, const Dims & new_dims)`

Like [downSize\(\)](#), except image proportions don't have to match exactly.

This operation applies alternating lowpass/decimate as long as the result is larger than the desired size, and then finally does a bilinear interpolation to the final size. If the desired size is actually larger than the input size, then the image is simply upscaled with a bilinear interpolation. The advantage of using this operation over just using [rescale\(\)](#) directly is that [rescale\(\)](#) may introduce aliasing if an image is being downsized by several ocatves; successive lowpass/decimate operations ensure that total image energy is preserved.

Definition at line 268 of file Image\_ShapeOps.C.

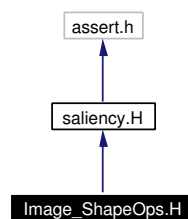
References `ASSERT`, `decX()`, `decY()`, `Image< T >::getDims()`, `Image< T >::getHeight()`, `Image< T >::getWidth()`, `Dims::h()`, `Dims::isNonEmpty()`, `lowPassX()`, `lowPassY()`, `rescale()`, and `Dims::w()`.

Referenced by `VisualCortex::getOutput()`.

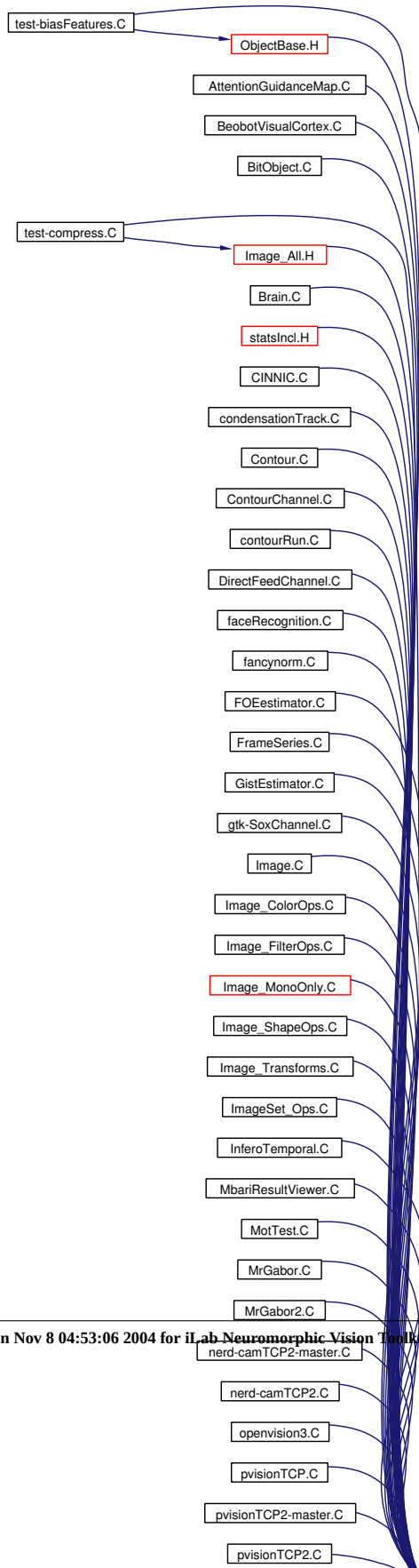
## 11.180 Image\_ShapeOps.H File Reference

```
#include "saliency.H"
```

Include dependency graph for Image\_ShapeOps.H:



This graph shows which files directly or indirectly include this file:



## Functions

- `template<class T_or_RGB> Image< T_or_RGB > quickLocalAvg (const Image< T_or_RGB > &array, const int scale)`  
*Compute quick and dirty scaling down of array.*
- `template<class T_or_RGB> Image< T_or_RGB > quickLocalMax (const Image< T_or_RGB > &array, const int scale)`  
*Compute quick and dirty scaling down of array.*
- `template<class T_or_RGB> Image< T_or_RGB > quickLocalMin (const Image< T_or_RGB > &array, const int scale)`  
*Compute quick and dirty scaling down of array.*
- `template<class T_or_RGB> Image< T_or_RGB > quickInterpolate (const Image< T_or_RGB > &src, const int sfactor)`  
*Dumb interpolation.*
- `template<class T_or_RGB> Image< T_or_RGB > rescale (const Image< T_or_RGB > &src, const Dims &dims)`  
*Scale to new width & height using bilinear interpolation.*
- `template<class T_or_RGB> Image< T_or_RGB > rescale (const Image< T_or_RGB > &src, int width, int height)`  
*Scale to new width & height using bilinear interpolation.*
- `template<class T> Image< T > downSize (const Image< T > &src, const Dims &dims)`  
*Downsize using alternating lowpass/decimate operations.*
- `template<class T> Image< T > downSize (const Image< T > &src, const int width, const int height)`  
*Downsize using alternating lowpass/decimate operations.*
- `Image< float > downSizeClean (const Image< float > &src, const Dims &new_dims)`  
*Like `downSize()`, except image proportions don't have to match exactly.*
- `template<class T_or_RGB> Image< T_or_RGB > concatX (const Image< T_or_RGB > &left, const Image< T_or_RGB > &right)`  
*Juxtapose two arrays of same height into one.*
- `template<class T_or_RGB> Image< T_or_RGB > concatY (const Image< T_or_RGB > &top, const Image< T_or_RGB > &bottom)`



*Juxtapose two arrays of same width into one.*

- `template<class T_or_RGB> Image< T_or_RGB > concatArray (const Image< T_or_RGB > arr[ ], const int nbarr, const int Nx, const int destX=-1, const int destY=-1)`

*Concatenate all image arrays, Nx arrays per line.*

- `template<class T_or_RGB> Image< T_or_RGB > decXY (const Image< T_or_RGB > &src, const int factor=2)`

*Decimate in X and Y (take one every 'factor' pixels).*

- `template<class T_or_RGB> Image< T_or_RGB > decX (const Image< T_or_RGB > &src, const int factor=2)`

*Decimate in X (take one every 'factor' pixels).*

- `template<class T_or_RGB> Image< T_or_RGB > decY (const Image< T_or_RGB > &src, const int factor=2)`

*Decimate in Y (take one every 'factor' pixels).*

- `template<class T_or_RGB> Image< T_or_RGB > intXY (const Image< T_or_RGB > &src, const bool dupli)`

*Interpolate in X and Y (zero-pad if dupli == false, or duplicate).*

- `template<class T_or_RGB> Image< T_or_RGB > intX (const Image< T_or_RGB > &src, const bool dupli)`

*Interpolate in X (zero-pad if dupli == false or duplicate).*

- `template<class T_or_RGB> Image< T_or_RGB > intY (const Image< T_or_RGB > &src, const bool dupli)`

*Interpolate in Y (zero-pad if dupli == false or duplicate).*

- `template<class T_or_RGB> Image< T_or_RGB > zoomXY (const Image< T_or_RGB > &src, int xzoom=2, int yzoom=-1)`

*Zoom up an image by duplicating pixels.*

- `template<class T_or_RGB> Image< T_or_RGB > crop (const Image< T_or_RGB > &src, const Point2D &upperLeft, const Dims &size, const bool zeroFill=false)`

*Extract a subimage from the input image.*

- `template<class T_or_RGB> Image< T_or_RGB > crop (const Image< T_or_RGB > &src, const Rectangle &rect, const bool zeroFill=false)`

*Overload of `crop()` using a `Rectangle` instead of a `Point2D+Dims`.*

- `template<class T_or_RGB> Image< T_or_RGB > shift (const Image< T_or_RGB > &srcImg, const int dx, const int dy)`

*Shift an image by (int dx, int dy), with horizontal wraparound.*

- `template<class T_or_RGB> Image< T_or_RGB > shiftImage (const Image< T_or_RGB > &srcImg, const float dx, const float dy)`

*Shift an image by (float dx, float dy).*

- `template<class T_or_RGB> Image< T_or_RGB > shiftClean (const Image< T_or_RGB > &srcImg, const int dx, const int dy, const T_or_RGB bgval=T_or_RGB())`

*Shift an image by (int dx, int dy), without wraparound.*

- `template<class T_or_RGB> Image< T_or_RGB > rotate (const Image< T_or_RGB > &srcImg, const int x, const int y, const float ang)`

*Rotate an image about (x,y) by ang(in Radians), without interpolation.*

- `template<class T_or_RGB> Image< T_or_RGB > combineImagesVertically (const Image< T_or_RGB > &topImg, const Image< T_or_RGB > &bottomImg)`

*Returns an image consisting of topImg above bottomImg.*

### 11.180.1 Detailed Description

Shape operations on [Image](#)

Definition in file [Image\\_ShapeOps.H](#).

### 11.180.2 Function Documentation

- #### 11.180.2.1
- `template<class T_or_RGB> Image<T_or_RGB>  
combineImagesVertically (const Image< T_or_RGB > & topImg,  
const Image< T_or_RGB > & bottomImg)`

Returns an image consisting of topImg above bottomImg.

Referenced by `VisualObject::match()`.

**11.180.2.2** `template<class T_or_RGB> Image<T_or_RGB> concatArray  
(const Image< T_or_RGB > arr[ ], const int nbarr, const int Nx,  
const int destX = -1, const int destY = -1)`

Concatenate all image arrays, Nx arrays per line.

Also reshape each array to (destX, destY) if these are != -1.

Referenced by `main()`, `makeImageArray()`, and `Raster::VisuArray()`.

**11.180.2.3** `template<class T_or_RGB> Image<T_or_RGB> concatX (const  
Image< T_or_RGB > & left, const Image< T_or_RGB > & right)`

Juxtapose two arrays of same height into one.

Referenced by `SimulationViewerStd::drawMotionSixPack()`, `SimulationViewerEyeMvt2::getTraj()`, `SimulationViewerEyeMvt::getTraj()`, `SimulationViewerCompress::getTraj()`, and `main()`.

**11.180.2.4** `template<class T_or_RGB> Image<T_or_RGB> concatY (const  
Image< T_or_RGB > & top, const Image< T_or_RGB > & bottom)`

Juxtapose two arrays of same width into one.

Referenced by `SimulationViewerStd::drawMotionSixPack()`, `SimulationViewerEyeMvt2::getTraj()`, `SimulationViewerCompress::getTraj()`, and `main()`.

**11.180.2.5** `template<class T_or_RGB> Image<T_or_RGB> crop (const  
Image< T_or_RGB > & src, const Rectangle & rect, const bool  
zerofill = false)`

Overload of `crop()` using a `Rectangle` instead of a `Point2D+Dims`.

Referenced by `BitObject::doesIntersect()`, `RetinaStd::doGetOutput()`, `dogFilterHmax()`, `TemplateMatchChannel::findBestMatch()`, `ShapeEstimator::getObjectImage()`, `SimulationViewerStd::getTraj()`, `SimulationViewerCompress::getTraj()`, `infoFFT()`, `BitObject::reset()`, `MbariResultViewer::saveSingleEventFrame()`, and `FOEestimator::updateFOE()`.

**11.180.2.6** `template<class T_or_RGB> Image<T_or_RGB> crop (const  
Image< T_or_RGB > & src, const Point2D & upperLeft, const Dims  
& size, const bool zerofill = false)`

Extract a subimage from the input image.

The upper left corner of the subimage is given by a [Point2D](#), and the size of the subimage is given by the [Dims](#). If zerofill is false, then the rectangle given by Point2D+Dims must be wholly contained within the bounds of the input image. If zerofill is true, then any output region that lies outside the bounds of the input will be zero-filled (where zero means the default value for type T).

**11.180.2.7** `template<class T_or_RGB> Image<T_or_RGB> decX (const Image< T_or_RGB > & src, const int factor = 2)`

Decimate in X (take one every 'factor' pixels).

Referenced by TemplateMatchPyrBuilder::build(), downSize(), and downSizeClean().

**11.180.2.8** `template<class T_or_RGB> Image<T_or_RGB> decXY (const Image< T_or_RGB > & src, const int factor = 2)`

Decimate in X and Y (take one every 'factor' pixels).

Referenced by SimulationViewerEyeMvt::getTraj(), SimulationViewerCompress::getTraj(), main(), and reduce().

**11.180.2.9** `template<class T_or_RGB> Image<T_or_RGB> decY (const Image< T_or_RGB > & src, const int factor = 2)`

Decimate in Y (take one every 'factor' pixels).

Referenced by TemplateMatchPyrBuilder::build(), downSize(), and downSizeClean().

**11.180.2.10** `template<class T> Image<T> downSize (const Image< T > & src, const int width, const int height)`

Downsize using alternating lowpass/decimate operations.

Definition at line 244 of file Image\_ShapeOps.C.

References ASSERT, decX(), decY(), Image< T >::getHeight(), Image< T >::getWidth(), lowPassX(), and lowPassY().

Referenced by Image< T >::addDownsized(), computeCMAP(), InferoTemporal::computeVisualObject(), downSize(), SimulationViewerCompress::getTraj(), maxNormalizeFancy(), and SingleChannel::postProcessMap().

### 11.180.2.11 `template<class T> Image<T> downSize (const Image< T > & src, const Dims & dims)`

Downsize using alternating lowpass/decimate operations.

Definition at line 237 of file Image\_ShapeOps.C.

References `downSize()`, `Dims::h()`, and `Dims::w()`.

### 11.180.2.12 `Image<float> downSizeClean (const Image< float > & src, const Dims & new_dims)`

Like `downSize()`, except image proportions don't have to match exactly.

This operation applies alternating lowpass/decimate as long as the result is larger than the desired size, and then finally does a bilinear interpolation to the final size. If the desired size is actually larger than the input size, then the image is simply upscaled with a bilinear interpolation. The advantage of using this operation over just using `rescale()` directly is that `rescale()` may introduce aliasing if an image is being downsized by several octaves; successive lowpass/decimate operations ensure that total image energy is preserved.

Definition at line 268 of file Image\_ShapeOps.C.

References `ASSERT`, `decX()`, `decY()`, `Image< T >::getDims()`, `Image< T >::getHeight()`, `Image< T >::getWidth()`, `Dims::h()`, `Dims::isEmpty()`, `lowPassX()`, `lowPassY()`, `rescale()`, and `Dims::w()`.

Referenced by `VisualCortex::getOutput()`.

### 11.180.2.13 `template<class T_or_RGB> Image<T_or_RGB> intX (const Image< T_or_RGB > & src, const bool dupli)`

Interpolate in X (zero-pad if `dupli == false` or duplicate).

### 11.180.2.14 `template<class T_or_RGB> Image<T_or_RGB> intXY (const Image< T_or_RGB > & src, const bool dupli)`

Interpolate in X and Y (zero-pad if `dupli == false`, or duplicate).

Referenced by `main()`.

### 11.180.2.15 `template<class T_or_RGB> Image<T_or_RGB> intY (const Image< T_or_RGB > & src, const bool dupli)`

Interpolate in Y (zero-pad if `dupli == false` or duplicate).

**11.180.2.16** `template<class T_or_RGB> Image<T_or_RGB>  
quickInterpolate (const Image< T_or_RGB > & src, const int  
sfactor)`

Dumb interpolation.

**11.180.2.17** `template<class T_or_RGB> Image<T_or_RGB> quickLocalAvg  
(const Image< T_or_RGB > & array, const int scale)`

Compute quick and dirty scaling down of array.

Referenced by main().

**11.180.2.18** `template<class T_or_RGB> Image<T_or_RGB> quickLocalMax  
(const Image< T_or_RGB > & array, const int scale)`

Compute quick and dirty scaling down of array.

**11.180.2.19** `template<class T_or_RGB> Image<T_or_RGB> quickLocalMin  
(const Image< T_or_RGB > & array, const int scale)`

Compute quick and dirty scaling down of array.

**11.180.2.20** `template<class T_or_RGB> Image<T_or_RGB> rescale (const  
Image< T_or_RGB > & src, int width, int height)`

Scale to new width & height using bilinear interpolation.

**11.180.2.21** `template<class T_or_RGB> Image<T_or_RGB> rescale (const  
Image< T_or_RGB > & src, const Dims & dims)`

Scale to new width & height using bilinear interpolation.

**11.180.2.22** `template<class T_or_RGB> Image<T_or_RGB> rotate (const  
Image< T_or_RGB > & srcImg, const int x, const int y, const float  
ang)`

Rotate an image about (x,y) by ang(in Radians), without interpolation.

**11.180.2.23** `template<class T_or_RGB> Image<T_or_RGB> shift (const Image< T_or_RGB > & srcImg, const int dx, const int dy)`

Shift an image by (int dx, int dy), with horizontal wraparound.

**11.180.2.24** `template<class T_or_RGB> Image<T_or_RGB> shiftClean (const Image< T_or_RGB > & srcImg, const int dx, const int dy, const T_or_RGB bgval = T_or_RGB())`

Shift an image by (int dx, int dy), without wraparound.

Referenced by RetinaStd::doGetOutput().

**11.180.2.25** `template<class T_or_RGB> Image<T_or_RGB> shiftImage (const Image< T_or_RGB > & srcImg, const float dx, const float dy)`

Shift an image by (float dx, float dy).

Referenced by ReichardtPyrBuilder< T >::build(), DisparityChannel::doInput(), and main().

**11.180.2.26** `template<class T_or_RGB> Image<T_or_RGB> zoomXY (const Image< T_or_RGB > & src, int xzoom = 2, int yzoom = -1)`

Zoom up an image by duplicating pixels.

Separate scaling factors can be given for the x and y dimensions.

**Parameters:**

*src* Input image.

*xzoom* Zoom factor for x dimension (default 2).

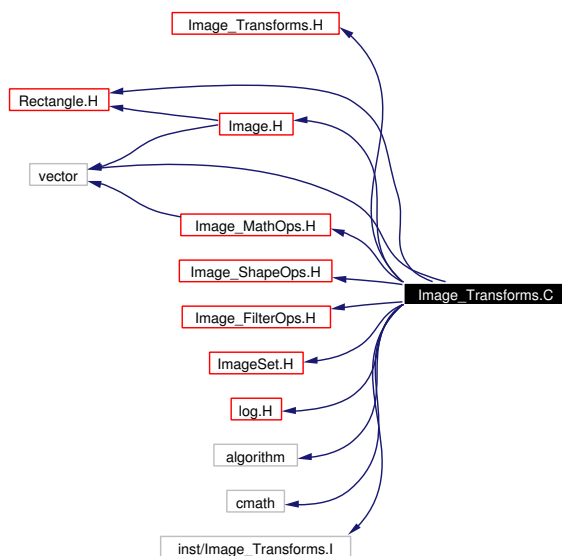
*yzoom* Zoom factor for y dimension. If negative (the default), then yzoom is taken to be the same as xzoom.

Referenced by makeImageArray().

## 11.181 Image\_Transforms.C File Reference

```
#include "Image_Transforms.H"
#include "Image.H"
#include "Image_MathOps.H"
#include "Image_ShapeOps.H"
#include "Image_FilterOps.H"
#include "ImageSet.H"
#include "log.H"
#include "Rectangle.H"
#include <algorithm>
#include <vector>
#include <cmath>
#include "inst/Image_Transforms.I"
```

Include dependency graph for Image\_Transforms.C:



## Defines

- #define **SEGOBJ\_START** 0.9
- #define **SEGOBJ\_STEP** 0.05



- #define **SEGOBJ\_MINA** 0.005
- #define **SEGOBJ\_EXPL** 1.05

## Functions

- template<class T> **Image**< **byte** > **segmentObject** (const **Image**< T > &src, const **Point2D** &seed)  
*Segment object using an adaptive thresholding & region growing method.*
- template<class T> int **segmentLandmark** (const **Image**< T > &src, const **Point2D** &seed, **Image**< **byte** > &target, double &activity, double &standout, float &area\_percentage)  
*Segment landmark using an adaptive thresholding & region growing method.*
- template<class T> **Image**< **byte** > **contour2D** (const **Image**< T > &src)  
*Compute 2D 4-neighbor contours of binary objects in image.*
- template<class T> **Image**< **byte** > **contour2D8** (const **Image**< T > &src)  
*Compute 2D 8-neighbor contours of binary objects in image.*
- template<class T> **Image**< T > **chamfer34** (const **Image**< T > &src, const T dmax)  
*3/4-chamfer distance transform; object = non 0*
- template<class T> **Image**< typename **promote\_trait**< T, float >::TP > **dct** (const **Image**< T > &src, const int offx, const int offy, const int size)  
*Computes 2D discrete cosine transform.*
- template<class T> float **infoFFT** (const **Image**< T > &src, const float eps, const **Rectangle** &rect)  
*FFT-based information measure.*
- template<class T> **Image**< float > **infoFFT** (const **Image**< T > &src, const float eps)  
*FFT-based information measure.*
- double **learningCoeff** (const **Image**< float > &featureMap, const **Image**< **byte** > &targetDMap, const bool softmax, const int in\_thresh, const int out\_thresh)  
*Examine local maxes in/out of targets and compute a learning weight.*
- template<class T> **Image**< T > **scaleBlock** (const **Image**< T > &src, const **Dims** newDims)

- `template<class T> int floodClean (const Image< T > &src, Image< T > &dest, const Point2D &seed, const T thresh, const T val)`  
*Floods object >= thresh with val from seed and returns area.*
- `template<class T> int floodCleanBB (const Image< T > &src, Image< T > &dest, const Point2D &seed, const T thresh, const T val, Rectangle &bbox)`  
*same as floodClean, only finds the bounding box of the region in addition*
- `template<class T> Image< float > distDegrade (const Image< T > &src, const Point2D &foa, const float area)`  
*degrades a binary image mask s.t. it decays away from the FOA*
- `template<class T> Image< byte > segmentObjectClean (const Image< T > &src, const Point2D &seed)`  
*same as segmentObject, only with floodClean and different parameters*
- `template<class T> Image< T > highThresh (const Image< T > &src, const T thresh, const T val)`  
*set all values >= thresh to val in src and return the result*
- `template<class T> Image< T > foveate (const Image< byte > &mask, const ImageSet< T > &pyr)`
- `template<class T> Image< T > weightedBlur (const Image< byte > &modulator, const ImageSet< T > &pyr)`
- `template<class T_or_RGB> Image< T_or_RGB > replaceVals (const Image< T_or_RGB > &src, const T_or_RGB from, const T_or_RGB to, const T_or_RGB other)`  
*replace all values "from" with "to" and all other values with "other"*
- `template<class T_or_RGB> Image< T_or_RGB > replaceVals (const Image< T_or_RGB > &src, const T_or_RGB from, const T_or_RGB to)`  
*replace all values "from" with "to" and leave all other values in place*
- `template<class T> Image< byte > makeBinary (const Image< T > &src, const T &threshold, const byte lowVal, const byte highVal)`  
*make a binary version of the source image*
- `template<class T_or_RGB> void pasteImage (Image< T_or_RGB > &background, const Image< T_or_RGB > &foreground, const T_or_RGB &transparent, const Point2D location, float opacity)`  
*paste foreground into background image at (location)*
- `template<class T_or_RGB> Image< T_or_RGB > copy (const Image< T_or_RGB > &src, const Rectangle &r)`

*return a Copy a window of image passed in*

### 11.181.1 Detailed Description

Transformations on [Image](#)

Definition in file [Image\\_Transforms.C](#).

### 11.181.2 Function Documentation

#### 11.181.2.1 `template<class T> Image<T> chamfer34 (const Image< T > & src, const T dmax)`

3/4-chamfer distance transform; object = non 0

Definition at line 269 of file [Image\\_Transforms.C](#).

References [ASSERT](#), [Image](#)< T >::begin(), [Image](#)< T >::beginw(), [Image](#)< T >::getDims(), [Image](#)< T >::getHeight(), [Image](#)< T >::getSize(), [Image](#)< T >::getWidth(), [Dims](#)::h(), [Image](#)< T >::initialized(), and [Dims](#)::w().

Referenced by [VisualBuffer::findMostInterestingTarget\(\)](#), [ShapeEstimator::getSmoothMask\(\)](#), [TaskRelevanceMapKillN::input\(\)](#), [VisualBuffer::input\(\)](#), and [VisualCortex::learn\(\)](#).

#### 11.181.2.2 `template<class T> Image<byte> contour2D (const Image< T > & src)`

Compute 2D 4-neighbor contours of binary objects in image.

Definition at line 202 of file [Image\\_Transforms.C](#).

References [ASSERT](#), [Image](#)< T >::begin(), [Image](#)< T >::beginw(), [Image](#)< T >::getDims(), [Image](#)< T >::getHeight(), [Image](#)< T >::getWidth(), [Image](#)< T >::initialized(), and [NO\\_INIT](#).

#### 11.181.2.3 `template<class T> Image<byte> contour2D8 (const Image< T > & src)`

Compute 2D 8-neighbor contours of binary objects in image.

Definition at line 233 of file [Image\\_Transforms.C](#).

References [ASSERT](#), [Image](#)< T >::begin(), [Image](#)< T >::beginw(), [Image](#)< T >::getDims(), [Image](#)< T >::getHeight(), [Image](#)< T >::getWidth(), [Image](#)< T >::initialized(), and [NO\\_INIT](#).

Referenced by `SimulationViewerStd::drawMaskOutline()`, `SimulationViewerCompress::getTraj()`, and `SimulationViewerEyeMvt2::getTraj()`.

#### 11.181.2.4 `template<class T_or_RGB> Image<T_or_RGB> copy (const Image< T_or_RGB > & src, const Rectangle & r)`

return a Copy a window of image passed in

Definition at line 1337 of file `Image_Transforms.C`.

References `ASSERT`, `Image< T >::beginw()`, `Rectangle::bottom()`, `Image< T >::getVal()`, `Image< T >::initialized()`, `Rectangle::left()`, `Image< T >::rectangleOk()`, `Rectangle::right()`, `Rectangle::top()`, and `ZEROS`.

Referenced by `safecopy()`, and `ArrayHandle< T >::uniq()`.

#### 11.181.2.5 `template<class T> Image<typename promote_trait<T, float>::TP> dct (const Image< T > & src, const int offx, const int offy, const int size)`

Computes 2D discrete cosine transform.

of square image patch size\*size at offset (offx, offy)

Definition at line 389 of file `Image_Transforms.C`.

References `ASSERT`, `Image< T >::getHeight()`, `Image< T >::getVal()`, `Image< T >::getWidth()`, `Image< T >::initialized()`, `NO_INIT`, and `Image< T >::setVal()`.

Referenced by `infoMeasure()`.

#### 11.181.2.6 `template<class T> Image<float> distDegrade (const Image< T > & src, const Point2D & fof, const float area)`

degrades a binary image mask s.t. it decays away from the FOA

Definition at line 1095 of file `Image_Transforms.C`.

References `Image< T >::begin()`, `Image< T >::beginw()`, `exp()`, `Image< T >::getDims()`, `Image< T >::getHeight()`, `Image< T >::getWidth()`, `Point2D::i`, `Point2D::j`, and `ZEROS`.

#### 11.181.2.7 `template<class T> int floodClean (const Image< T > & src, Image< T > & dest, const Point2D & seed, const T thresh, const T val)`

Floods object  $\geq$  thresh with val from seed and returns area.

a clean version of [Image::flood](#) - problems solved: correct boundary checking, allow for thresh = 0.

**Parameters:**

**dest** if dest is not initialized, it will be initialized to ZEROS. ATTENTION: If you want to have the old behavior of dest being initialized to src, you have to this explicitly before calling floodClean

Definition at line 976 of file Image\_Transforms.C.

References ASSERT, Image< T >::coordsOk(), Image< T >::getDims(), Image< T >::getVal(), Image< T >::initialized(), Image< T >::resize(), Image< T >::setVal(), and ZEROS.

Referenced by main(), and segmentObjectClean().

**11.181.2.8** `template<class T> int floodCleanBB (const Image< T > & src, Image< T > & dest, const Point2D & seed, const T thresh, const T val, Rectangle & bbox)`

same as floodClean, only finds the bounding box of the region in addition

Definition at line 1031 of file Image\_Transforms.C.

References ASSERT, Image< T >::coordsOk(), Image< T >::getDims(), Image< T >::getHeight(), Image< T >::getVal(), Image< T >::getWidth(), Point2D::i, Image< T >::initialized(), Point2D::j, Rectangle::modify(), Image< T >::resize(), Image< T >::setVal(), and ZEROS.

Referenced by BitObject::reset().

**11.181.2.9** `template<class T> Image<T> highThresh (const Image< T > & src, const T thresh, const T val)`

set all values >= thresh to val in src and return the result

Definition at line 1165 of file Image\_Transforms.C.

References ASSERT, Image< T >::beginw(), Image< T >::endw(), and Image< T >::initialized().

Referenced by EntropyChannel::doInput().

**11.181.2.10** `template<class T> Image<float> infoFFT (const Image< T > & src, const float eps)`

FFT-based information measure.

Definition at line 733 of file Image\_Transforms.C.

References `Image< T >::getDims()`, `Image< T >::getHeight()`, `Image< T >::getVal()`, `Image< T >::getWidth()`, `Dims::h()`, `NO_INIT`, `Image< T >::setVal()`, and `Dims::w()`.  
Referenced by `infoFFT()`.

#### 11.181.2.11 `template<class T> float infoFFT (const Image< T > & src, const float eps, const Rectangle & rect)`

FFT-based information measure.

Definition at line 722 of file `Image_Transforms.C`.

#### 11.181.2.12 `double learningCoeff (const Image< float > & featureMap, const Image< byte > & targetDMap, const bool softmax = false, const int in_thresh = 20, const int out_thresh = 255)`

Examine local maxes in/out of targets and compute a learning weight.

Valid ranges for `in_thresh` and `out_thresh` are 0..255, plus: use -1 for `in_thresh` to learn from the background only, or use -1 for `out_thresh` to learn from the target only.

Definition at line 881 of file `Image_Transforms.C`.

References `ASSERT`, `Image< T >::begin()`, `binaryReverse()`, `byte`, `Image< T >::end()`, `Image< T >::getSize()`, `Image< T >::initialized()`, `Image< T >::isSameSize()`, `mean()`, and `sum()`.

Referenced by `SingleChannel::learn()`.

#### 11.181.2.13 `template<class T> Image<byte> makeBinary (const Image< T > & src, const T & threshold, const byte lowVal = byte(0), const byte highVal = byte(255))`

make a binary version of the source image

##### Parameters:

**src** source image

**threshold** the threshold used for binarization

**lowVal** value for all points  $\leq$  threshold

**highVal** value for all points  $>$  threshold

Definition at line 1280 of file `Image_Transforms.C`.

References `Image< T >::begin()`, `Image< T >::beginw()`, `Image< T >::endw()`, `Image< T >::getDims()`, and `NO_INIT`.

Referenced by `VisualBuffer::findMostInterestingTarget()`, and `main()`.

**11.181.2.14** `template<class T_or_RGB> void pasteImage (Image< T_or_RGB > & background, const Image< T_or_RGB > & foreground, const T_or_RGB & transparent, const Point2D location, float opacity = 1.0F)`

paste foreground into background image at (location)

**Parameters:**

*transparent* pixels with this value are not replaced in background

Definition at line 1293 of file Image\_Transforms.C.

References Image< T >::begin(), Image< T >::beginw(), Image< T >::getHeight(), Image< T >::getWidth(), Point2D::i, and Point2D::j.

Referenced by VisualEventSet::drawTokens(), BitObject::getObjectMask(), and VisualBuffer::input().

**11.181.2.15** `template<class T_or_RGB> Image<T_or_RGB> replaceVals (const Image< T_or_RGB > & src, const T_or_RGB from, const T_or_RGB to)`

replace all values "from" with "to" and leave all other values in place

Definition at line 1265 of file Image\_Transforms.C.

References Image< T >::begin(), Image< T >::beginw(), Image< T >::endw(), Image< T >::getDims(), and NO\_INIT.

Referenced by VisualEventSet::drawTokens(), extractBitObjects(), and BitObject::getObjectMask().

**11.181.2.16** `template<class T_or_RGB> Image<T_or_RGB> replaceVals (const Image< T_or_RGB > & src, const T_or_RGB from, const T_or_RGB to, const T_or_RGB other)`

replace all values "from" with "to" and all other values with "other"

Definition at line 1250 of file Image\_Transforms.C.

**11.181.2.17** `template<class T> int segmentLandmark (const Image< T > & src, const Point2D & seed, Image< byte > & target, double & activity, double & standout, float & area_percentage)`

Segment landmark using an adaptive thresholding & region growing method.

Returns an image that masks the landmark

**Parameters:**

**activity** is a measure of mean activity in the landmark

**standout** is a measure of how much the landmark stands out from the rest of the image

**area** is the percentage area of landmark

Definition at line 97 of file Image\_Transforms.C.

References ASSERT, Image< T >::flood(), Image< T >::getSize(), Image< T >::getVal(), Point2D::i, Image< T >::initialized(), and Point2D::j.

Referenced by findPeaks(), goodness\_map(), and maxNormalizeLandmark().

#### 11.181.2.18 `template<class T> Image<byte> segmentObject (const Image< T > & src, const Point2D & center)`

Segment object using an adaptive thresholding & region growing method.

Returns an image that masks the object (see Itti et al, J Neuroimaging, 2001)

Definition at line 61 of file Image\_Transforms.C.

References ASSERT, Image< T >::flood(), Image< T >::getSize(), Image< T >::getVal(), Point2D::i, Image< T >::initialized(), and Point2D::j.

#### 11.181.2.19 `template<class T> Image<byte> segmentObjectClean (const Image< T > & src, const Point2D & seed)`

same as segmentObject, only with floodClean and different parameters

Definition at line 1124 of file Image\_Transforms.C.

References ASSERT, floodClean(), Image< T >::freeMem(), Image< T >::getSize(), Image< T >::getVal(), Point2D::i, Image< T >::initialized(), and Point2D::j.

Referenced by SimulationViewerEyeMvt2::eyeMovement(), and ShapeEstimator::getObjectMask().

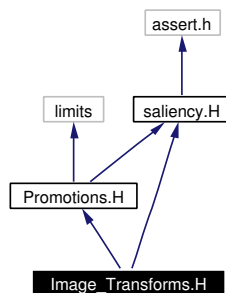


## 11.182 Image\_Transforms.H File Reference

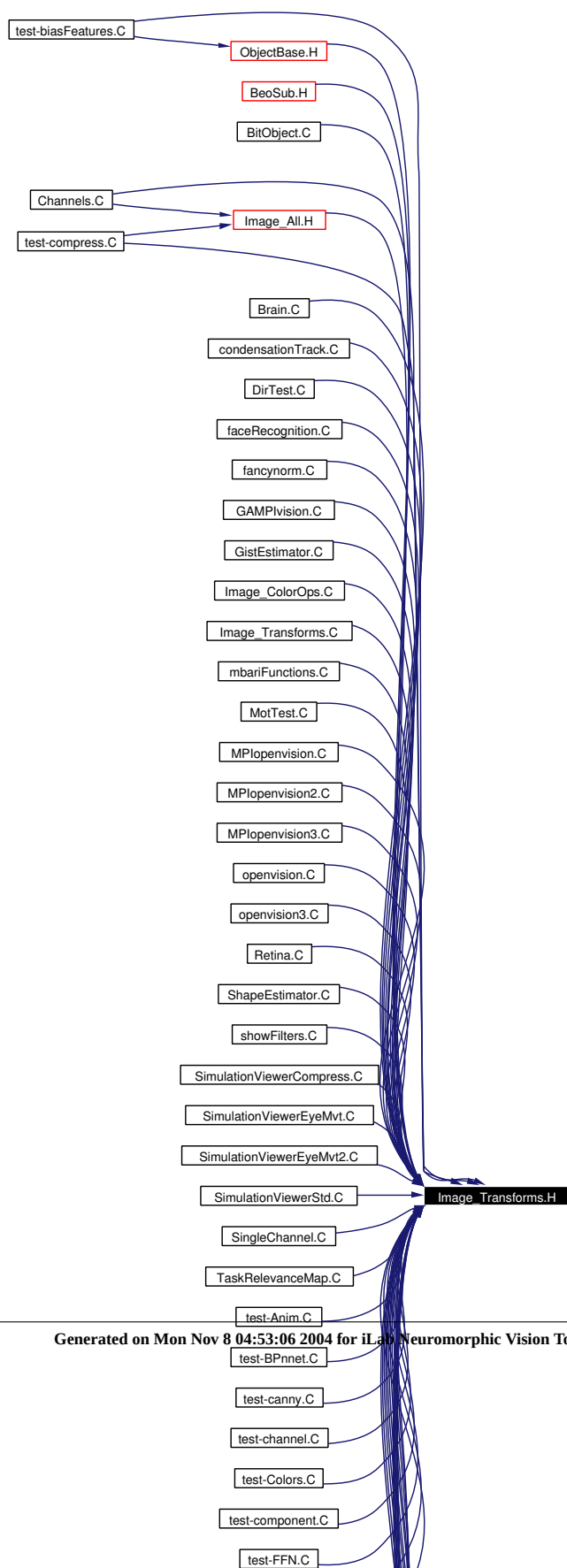
```
#include "saliency.H"
```

```
#include "Promotions.H"
```

Include dependency graph for Image\_Transforms.H:



This graph shows which files directly or indirectly include this file:



## Functions

- `template<class T> Image< byte > segmentObject (const Image< T > &src, const Point2D &center)`  
*Segment object using an adaptive thresholding & region growing method.*
- `template<class T> int segmentLandmark (const Image< T > &src, const Point2D &seed, Image< byte > &target, double &activity, double &standout, float &area_percentage)`  
*Segment landmark using an adaptive thresholding & region growing method.*
- `template<class T> Image< byte > contour2D (const Image< T > &src)`  
*Compute 2D 4-neighbor contours of binary objects in image.*
- `template<class T> Image< byte > contour2D8 (const Image< T > &src)`  
*Compute 2D 8-neighbor contours of binary objects in image.*
- `template<class T> Image< T > chamfer34 (const Image< T > &src, const T dmax=255)`  
*3/4-chamfer distance transform; object = non 0*
- `template<class T> Image< typename promote_trait< T, float >::TP > dct (const Image< T > &src, const int offx, const int offy, const int size)`  
*Computes 2D discrete cosine transform.*
- `template<class T> float infoFFT (const Image< T > &src, const float eps, const Rectangle &rect)`  
*FFT-based information measure.*
- `template<class T> Image< float > infoFFT (const Image< T > &src, const float eps)`  
*FFT-based information measure.*
- `double learningCoeff (const Image< float > &featureMap, const Image< byte > &targetDMap, const bool softmax=false, const int in_thresh=20, const int out_thresh=255)`  
*Examine local maxes in/out of targets and compute a learning weight.*
- `template<class T_or_RGB> Image< T_or_RGB > scaleBlock (const Image< T_or_RGB > &src, const Dims newDims)`  
*scale up an image using blocks*
- `template<class T> int floodClean (const Image< T > &src, Image< T > &dest, const Point2D &seed, const T thresh, const T val)`

*Floods object  $\geq$  thresh with val from seed and returns area.*

- `template<class T> int floodCleanBB (const Image< T > &src, Image< T > &dest, const Point2D &seed, const T thresh, const T val, Rectangle &bbox)`  
*same as floodClean, only finds the bounding box of the region in addition*
- `template<class T> Image< byte > segmentObjectClean (const Image< T > &src, const Point2D &seed)`  
*same as segmentObject, only with floodClean and different parameters*
- `template<class T> Image< float > distDegrade (const Image< T > &src, const Point2D &foa, const float area)`  
*degrades a binary image mask s.t. it decays away from the FOA*
- `template<class T> Image< T > highThresh (const Image< T > &src, const T thresh, const T val)`  
*set all values  $\geq$  thresh to val in src and return the result*
- `template<class T_or_RGB> Image< T_or_RGB > foveate (const Image< byte > &mask, const ImageSet< T_or_RGB > &pyr)`  
*Generate a foveation blur.*
- `template<class T_or_RGB> Image< T_or_RGB > replaceVals (const Image< T_or_RGB > &src, const T_or_RGB from, const T_or_RGB to, const T_or_RGB other)`  
*replace all values "from" with "to" and all other values with "other"*
- `template<class T_or_RGB> Image< T_or_RGB > replaceVals (const Image< T_or_RGB > &src, const T_or_RGB from, const T_or_RGB to)`  
*replace all values "from" with "to" and leave all other values in place*
- `template<class T_or_RGB> Image< T_or_RGB > weightedBlur (const Image< byte > &modulator, const ImageSet< T_or_RGB > &pyr)`  
*generate a weighted blur*
- `template<class T> Image< byte > makeBinary (const Image< T > &src, const T &threshold, const byte lowVal=byte(0), const byte highVal=byte(255))`  
*make a binary version of the source image*
- `template<class T_or_RGB> void pasteImage (Image< T_or_RGB > &background, const Image< T_or_RGB > &foreground, const T_or_RGB &transparent, const Point2D location, float opacity=1.0F)`  
*paste foreground into background image at (location)*

- `template<class T_or_RGB> Image< T_or_RGB > copy (const Image< T_or_RGB > &src, const Rectangle &r)`  
*return a Copy a window of image passed in*

### 11.182.1 Detailed Description

Transformations on [Image](#)

Definition in file [Image\\_Transforms.H](#).

### 11.182.2 Function Documentation

#### 11.182.2.1 `template<class T> Image<T> chamfer34 (const Image< T > &src, const T dmax = 255)`

3/4-chamfer distance transform; object = non 0

Definition at line 269 of file [Image\\_Transforms.C](#).

References [ASSERT](#), [Image< T >::begin\(\)](#), [Image< T >::beginw\(\)](#), [Image< T >::getDims\(\)](#), [Image< T >::getHeight\(\)](#), [Image< T >::getSize\(\)](#), [Image< T >::getWidth\(\)](#), [Dims::h\(\)](#), [Image< T >::initialized\(\)](#), and [Dims::w\(\)](#).

Referenced by [VisualBuffer::findMostInterestingTarget\(\)](#), [ShapeEstimator::getSmoothMask\(\)](#), [VisualBuffer::input\(\)](#), [TaskRelevanceMapKillN::input\(\)](#), and [VisualCortex::learn\(\)](#).

#### 11.182.2.2 `template<class T> Image<byte> contour2D (const Image< T > &src)`

Compute 2D 4-neighbor contours of binary objects in image.

Definition at line 202 of file [Image\\_Transforms.C](#).

References [ASSERT](#), [Image< T >::begin\(\)](#), [Image< T >::beginw\(\)](#), [Image< T >::getDims\(\)](#), [Image< T >::getHeight\(\)](#), [Image< T >::getWidth\(\)](#), [Image< T >::initialized\(\)](#), and [NO\\_INIT](#).

#### 11.182.2.3 `template<class T> Image<byte> contour2D8 (const Image< T > &src)`

Compute 2D 8-neighbor contours of binary objects in image.

Definition at line 233 of file [Image\\_Transforms.C](#).

References ASSERT, Image< T >::begin(), Image< T >::beginw(), Image< T >::getDims(), Image< T >::getHeight(), Image< T >::getWidth(), Image< T >::initialized(), and NO\_INIT.

Referenced by SimulationViewerStd::drawMaskOutline(), SimulationViewerEyeMvt2::getTraj(), and SimulationViewerCompress::getTraj().

#### 11.182.2.4 **template<class T\_or\_RGB> Image<T\_or\_RGB> copy (const Image< T\_or\_RGB > & src, const Rectangle & r)**

return a Copy a window of image passed in

Definition at line 1337 of file Image\_Transforms.C.

References ASSERT, Image< T >::beginw(), Rectangle::bottom(), Image< T >::getVal(), Image< T >::initialized(), Rectangle::left(), Image< T >::rectangleOk(), Rectangle::right(), Rectangle::top(), and ZEROS.

Referenced by safecopy(), and ArrayHandle< T >::uniq().

#### 11.182.2.5 **template<class T> Image<typename promote\_trait<T, float>::TP> dct (const Image< T > & src, const int offx, const int offy, const int size)**

Computes 2D discrete cosine transform.

of square image patch size\*size at offset (offx, offy)

Definition at line 389 of file Image\_Transforms.C.

References ASSERT, Image< T >::getHeight(), Image< T >::getVal(), Image< T >::getWidth(), Image< T >::initialized(), NO\_INIT, and Image< T >::setVal().

Referenced by infoMeasure().

#### 11.182.2.6 **template<class T> Image<float> distDegrade (const Image< T > & src, const Point2D & foa, const float area)**

degrades a binary image mask s.t. it decays away from the FOA

Definition at line 1095 of file Image\_Transforms.C.

References Image< T >::begin(), Image< T >::beginw(), exp(), Image< T >::getDims(), Image< T >::getHeight(), Image< T >::getWidth(), Point2D::i, Point2D::j, and ZEROS.

### 11.182.2.7 `template<class T> int floodClean (const Image< T > & src, Image< T > & dest, const Point2D & seed, const T thresh, const T val)`

Floods object  $\geq$  thresh with val from seed and returns area.

a clean version of `Image::flood` - problems solved: correct boundary checking, allow for thresh = 0.

#### Parameters:

**dest** if dest is not initialized, it will be initialized to ZEROS. ATTENTION: If you want to have the old behavior of dest being initialized to src, you have to this explicitly before calling floodClean

Definition at line 976 of file Image\_Transforms.C.

References ASSERT, Image< T >::coordsOk(), Image< T >::getDims(), Image< T >::getVal(), Image< T >::initialized(), Image< T >::resize(), Image< T >::setVal(), and ZEROS.

Referenced by main(), and segmentObjectClean().

### 11.182.2.8 `template<class T> int floodCleanBB (const Image< T > & src, Image< T > & dest, const Point2D & seed, const T thresh, const T val, Rectangle & bbox)`

same as floodClean, only finds the bounding box of the region in addition

Definition at line 1031 of file Image\_Transforms.C.

References ASSERT, Image< T >::coordsOk(), Image< T >::getDims(), Image< T >::getHeight(), Image< T >::getVal(), Image< T >::getWidth(), Point2D::i, Image< T >::initialized(), Point2D::j, Rectangle::modify(), Image< T >::resize(), Image< T >::setVal(), and ZEROS.

Referenced by BitObject::reset().

### 11.182.2.9 `template<class T_or_RGB> Image<T_or_RGB> foveate (const Image< byte > & mask, const ImageSet< T_or_RGB > & pyr)`

Generate a foveation blur.

Mask should be 255 inside the object and 0 outside; the pyramid should be a Gaussian pyramid and will be used to create the eccentricity-dependent blur. The deeper the pyramid, the more severe the blur

**11.182.2.10** `template<class T> Image<T> highThresh (const Image< T > & src, const T thresh, const T val)`

set all values  $\geq$  thresh to val in src and return the result

Definition at line 1165 of file Image\_Transforms.C.

References ASSERT, Image< T >::beginw(), Image< T >::endw(), and Image< T >::initialized().

Referenced by EntropyChannel::doInput().

**11.182.2.11** `template<class T> Image<float> infoFFT (const Image< T > & src, const float eps)`

FFT-based information measure.

Definition at line 733 of file Image\_Transforms.C.

References Image< T >::getDims(), Image< T >::getHeight(), Image< T >::getVal(), Image< T >::getWidth(), Dims::h(), NO\_INIT, Image< T >::setVal(), and Dims::w().

Referenced by infoFFT().

**11.182.2.12** `template<class T> float infoFFT (const Image< T > & src, const float eps, const Rectangle & rect)`

FFT-based information measure.

Definition at line 722 of file Image\_Transforms.C.

References crop(), Image< T >::getMinMax(), and infoFFT().

**11.182.2.13** `double learningCoeff (const Image< float > & featureMap, const Image< byte > & targetDMap, const bool softmax = false, const int in_thresh = 20, const int out_thresh = 255)`

Examine local maxes in/out of targets and compute a learning weight.

Valid ranges for in\_thresh and out\_thresh are 0..255, plus: use -1 for in\_thresh to learn from the background only, or use -1 for out\_thresh to learn from the target only.

Definition at line 881 of file Image\_Transforms.C.

References ASSERT, Image< T >::begin(), binaryReverse(), byte, Image< T >::end(), Image< T >::getSize(), Image< T >::initialized(), Image< T >::isSameSize(), mean(), and sum().

Referenced by SingleChannel::learn().



**11.182.2.14** `template<class T> Image<byte> makeBinary (const Image< T > & src, const T & threshold, const byte lowVal = byte(0), const byte highVal = byte(255))`

make a binary version of the source image

**Parameters:**

*src* source image

*threshold* the threshold used for binarization

*lowVal* value for all points <= threshold

*highVal* value for all points > threshold

Definition at line 1280 of file Image\_Transforms.C.

References `Image< T >::begin()`, `Image< T >::beginw()`, `Image< T >::endw()`, `Image< T >::getDims()`, and `NO_INIT`.

Referenced by `VisualBuffer::findMostInterestingTarget()`, and `main()`.

**11.182.2.15** `template<class T_or_RGB> void pasteImage (Image< T_or_RGB > & background, const Image< T_or_RGB > & foreground, const T_or_RGB & transparent, const Point2D location, float opacity = 1.0F)`

paste foreground into background image at (location)

**Parameters:**

*transparent* pixels with this value are not replaced in background

Definition at line 1293 of file Image\_Transforms.C.

References `Image< T >::begin()`, `Image< T >::beginw()`, `Image< T >::getHeight()`, `Image< T >::getWidth()`, `Point2D::i`, and `Point2D::j`.

Referenced by `VisualEventSet::drawTokens()`, `BitObject::getObjectMask()`, and `VisualBuffer::input()`.

**11.182.2.16** `template<class T_or_RGB> Image<T_or_RGB> replaceVals (const Image< T_or_RGB > & src, const T_or_RGB from, const T_or_RGB to)`

replace all values "from" with "to" and leave all other values in place

Definition at line 1265 of file Image\_Transforms.C.

References `Image< T >::begin()`, `Image< T >::beginw()`, `Image< T >::endw()`, `Image< T >::getDims()`, and `NO_INIT`.

Referenced by `VisualEventSet::drawTokens()`, `extractBitObjects()`, and `BitObject::getObjectMask()`.

**11.182.2.17** `template<class T_or_RGB> Image<T_or_RGB> replaceVals  
(const Image< T_or_RGB > & src, const T_or_RGB from, const  
T_or_RGB to, const T_or_RGB other)`

replace all values "from" with "to" and all other values with "other"

Definition at line 1250 of file `Image_Transforms.C`.

References `Image< T >::begin()`, `Image< T >::beginw()`, `Image< T >::endw()`, `Image< T >::getDims()`, and `NO_INIT`.

**11.182.2.18** `template<class T_or_RGB> Image<T_or_RGB> scaleBlock  
(const Image< T_or_RGB > & src, const Dims newDims)`

scale up an image using blocks

Referenced by `SimulationViewerStd::drawMotionSixPack()`, `ShapeEstimator::getSmoothMask()`, and `main()`.

**11.182.2.19** `template<class T> int segmentLandmark (const Image< T > &  
src, const Point2D & seed, Image< byte > & target, double &  
activity, double & standout, float & area_percentage)`

Segment landmark using an adaptive thresholding & region growing method.

Returns an image that masks the landmark

#### Parameters:

**activity** is a measure of mean activity in the landmark

**standout** is a measure of how much the landmark stands out from the rest of the image

**area** is the percentage area of landmark

Definition at line 97 of file `Image_Transforms.C`.

References `ASSERT`, `Image< T >::flood()`, `Image< T >::getSize()`, `Image< T >::getVal()`, `Point2D::i`, `Image< T >::initialized()`, and `Point2D::j`.

Referenced by `findPeaks()`, `goodness_map()`, and `maxNormalizeLandmark()`.

**11.182.2.20** `template<class T> Image<byte> segmentObject (const Image< T > & src, const Point2D & center)`

Segment object using an adaptive thresholding & region growing method.

Returns an image that masks the object (see Itti et al, J Neuroimaging, 2001)

Definition at line 61 of file Image\_Transforms.C.

References ASSERT, Image< T >::flood(), Image< T >::getSize(), Image< T >::getVal(), Point2D::i, Image< T >::initialized(), and Point2D::j.

**11.182.2.21** `template<class T> Image<byte> segmentObjectClean (const Image< T > & src, const Point2D & seed)`

same as segmentObject, only with floodClean and different parameters

Definition at line 1124 of file Image\_Transforms.C.

References ASSERT, floodClean(), Image< T >::freeMem(), Image< T >::getSize(), Image< T >::getVal(), Point2D::i, Image< T >::initialized(), and Point2D::j.

Referenced by SimulationViewerEyeMvt2::eyeMovement(), and ShapeEstimator::getObjectMask().

**11.182.2.22** `template<class T_or_RGB> Image<T_or_RGB> weightedBlur (const Image< byte > & modulator, const ImageSet< T_or_RGB > & pyr)`

generate a weighted blur

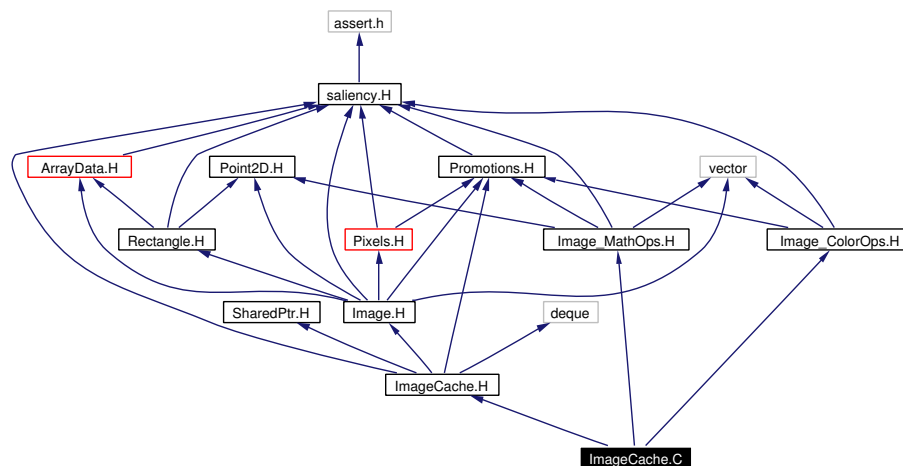
The higher the modulator value (0..255), the more blurring at that location

Referenced by SimulationViewerCompress::getTraj().

## 11.183 ImageCache.C File Reference

```
#include "ImageCache.H"
#include "Image_MathOps.H"
#include "Image_ColorOps.H"
```

Include dependency graph for ImageCache.C:



## Defines

- #define **INstantiate**(T)

### 11.183.1 Detailed Description

implements an image cache with running average

Definition in file [ImageCache.C](#).

### 11.183.2 Define Documentation

### 11.183.2.1 #define INSTANTIATE(T)

**Value:**

```
template class ImageCache< T >; \
        template class ImageCache< PixRGB< T > >; \
```

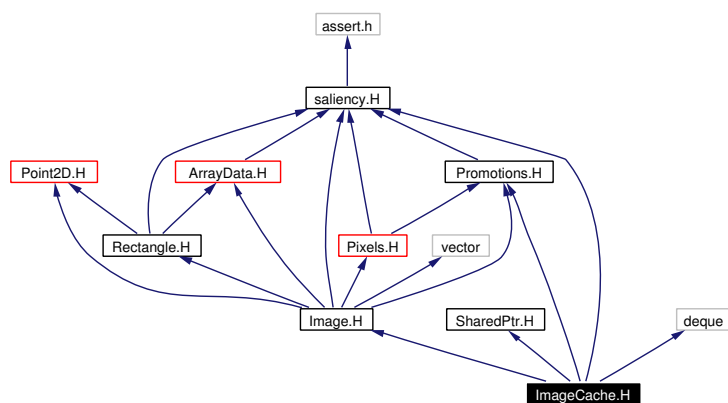
```
template class ImageCacheAvg< T >; \  
template class ImageCacheAvg< PixRGB< T > >; \  
template class ImageCacheMinMax< T >;
```

Definition at line 277 of file ImageCache.C.

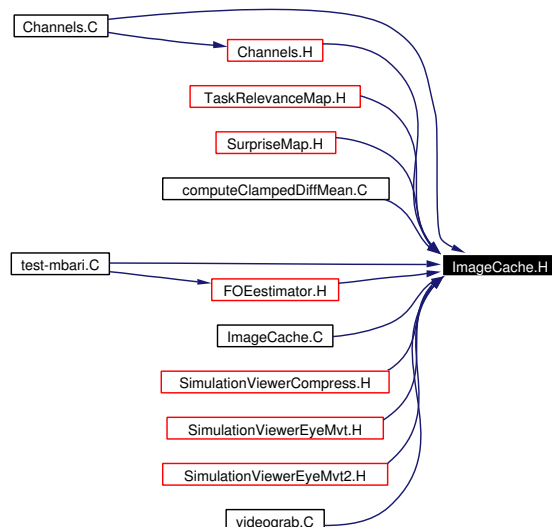
## 11.184 ImageCache.H File Reference

```
#include "saliency.H"  
#include "SharedPtr.H"  
#include "Image.H"  
#include "Promotions.H"  
#include <deque>
```

Include dependency graph for ImageCache.H:



This graph shows which files directly or indirectly include this file:



## Classes

- class [ImageCache](#)  
*base class for image caches that do computations on the fly*
- class [ImageCacheAvg](#)  
*image cache to compute the running average*
- class [ImageCacheMinMax](#)  
*image cache to compute a running min/max*

### 11.184.1 Detailed Description

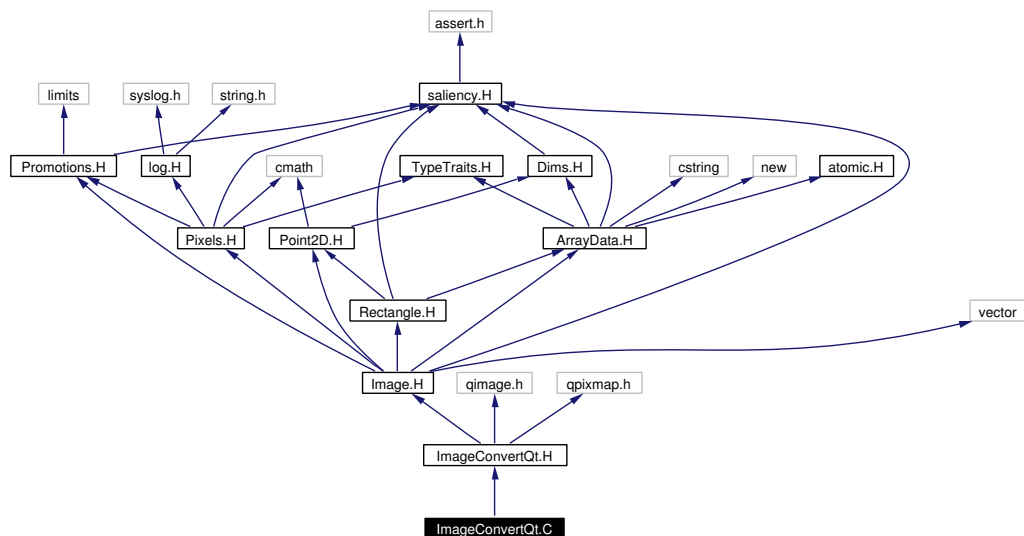
implements an image cache with running average

Definition in file [ImageCache.H](#).

## 11.185 ImageConvertQt.C File Reference

```
#include "ImageConvertQt.H"
```

Include dependency graph for ImageConvertQt.C:



### Functions

- QImage [convertToQImage](#) (Image< PixRGB< byte > > &img)

*convert Image< PixRGB<byte> > to QImage*

- QImage [convertToQImage](#) (Image< byte > &img)

*convert Image< byte > to QImage*

### 11.185.1 Detailed Description

Functions to convert our images to QImage or QPixmap

Definition in file [ImageConvertQt.C](#).



## 11.185.2 Function Documentation

### 11.185.2.1 QImage convertToQImage (Image< byte > & *img*)

convert Image< byte > to QImage

Definition at line 62 of file ImageConvertQt.C.

References Image< byte >::getArrayPtr(), Image< byte >::getHeight(), Image< byte >::getSize(), and Image< byte >::getWidth().

### 11.185.2.2 QImage convertToQImage (Image< PixRGB< byte > > & *img*)

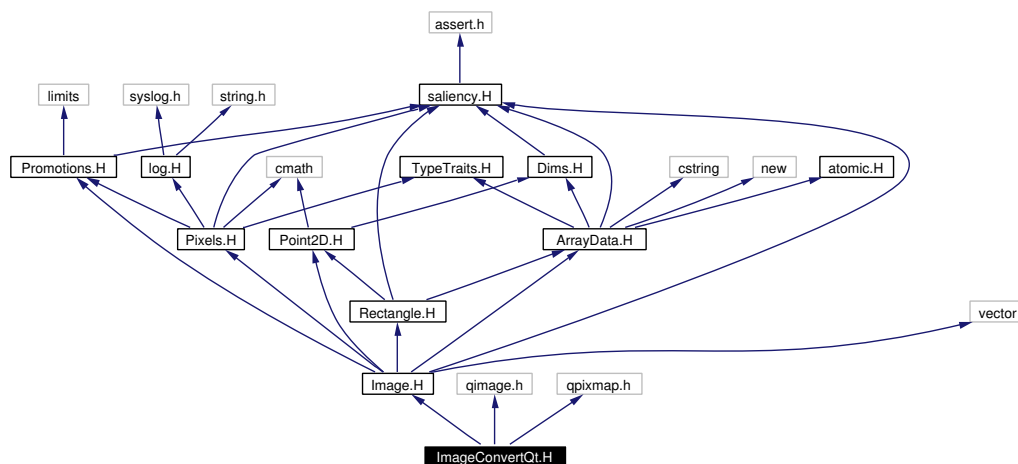
convert Image< PixRGB< byte > > to QImage

Definition at line 37 of file ImageConvertQt.C.

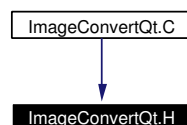
## 11.186 ImageConvertQt.H File Reference

```
#include "Image.H"
#include <qimage.h>
#include <qpixmap.h>
```

Include dependency graph for ImageConvertQt.H:



This graph shows which files directly or indirectly include this file:



## Functions

- QImage `convertToQImage` (Image< PixRGB< byte > > &img)  
convert Image< PixRGB<byte> > to QImage
- QImage `convertToQImage` (Image< byte > &img)  
convert Image< byte > to QImage

### 11.186.1 Detailed Description

Functions to convert our images to QImage or QPixmap

Definition in file [ImageConvertQt.H](#).

### 11.186.2 Function Documentation

#### 11.186.2.1 QImage convertToQImage ([Image](#)< [byte](#) > & *img*)

convert [Image](#)< [byte](#) > to QImage

Definition at line 62 of file ImageConvertQt.C.

References [Image](#)< [byte](#) >::getArrayPtr(), [Image](#)< [byte](#) >::getHeight(), [Image](#)< [byte](#) >::getSize(), and [Image](#)< [byte](#) >::getWidth().

#### 11.186.2.2 QImage convertToQImage ([Image](#)< [PixRGB](#)< [byte](#) > > & *img*)

convert [Image](#)< [PixRGB](#)<[byte](#)> > to QImage

Definition at line 37 of file ImageConvertQt.C.

## 11.187 ImageDisplay.C File Reference

```
#include "ImageDisplay.H"
```

```
#include "ModelOptionDefs.H"
```

```
#include "Raster.H"
```

Include dependency graph for ImageDisplay.C:

### 11.187.1 Detailed Description

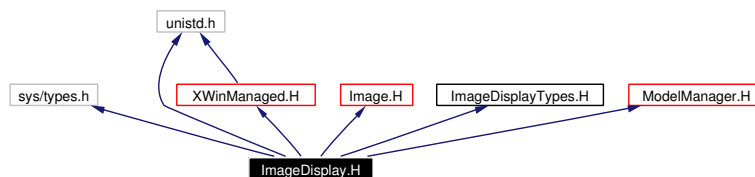
A generalized class for displaying images

Definition in file [ImageDisplay.C](#).

## 11.188 ImageDisplay.H File Reference

```
#include <sys/types.h>
#include <unistd.h>
#include "Image.H"
#include "ImageDisplayTypes.H"
#include "ModelManager.H"
#include "XWinManaged.H"
```

Include dependency graph for ImageDisplay.H:



This graph shows which files directly or indirectly include this file:



### Classes

- class [ImageDisplay](#)  
*A generalized class for displaying images.*

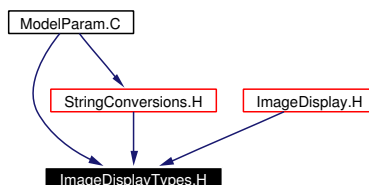
### 11.188.1 Detailed Description

A generalized class for displaying images

Definition in file [ImageDisplay.H](#).

## 11.189 ImageDisplayTypes.H File Reference

This graph shows which files directly or indirectly include this file:



### Defines

- `#define` [NBIMAGEDISPLAYTYPES](#) 4  
*number of known ImageDisplayTypes:*

### Enumerations

- `enum` [ImageDisplayType](#) { [DISPLAY\\_NONE](#) = 0, [DISPLAY\\_XV](#) = 1, [DISPLAY\\_XWIN](#) = 2, [DISPLAY\\_XWINMGD](#) = 3 }
- Valid values for ImageDisplayType (to customize [ImageDisplay](#)).*

### Functions

- `const char *` [imageDisplayTypeName](#) (`const` [ImageDisplayType](#) p)

#### 11.189.1 Detailed Description

Definition of types of [ImageDisplay](#)

Definition in file [ImageDisplayTypes.H](#).

#### 11.189.2 Define Documentation

##### 11.189.2.1 `#define` NBIMAGEDISPLAYTYPES 4

number of known ImageDisplayTypes:

Definition at line 46 of file [ImageDisplayTypes.H](#).

Referenced by `convertFromString()`.

### 11.189.3 Enumeration Type Documentation

#### 11.189.3.1 enum [ImageDisplayType](#)

Valid values for `ImageDisplayType` (to customize [ImageDisplay](#)).

**Enumeration values:**

***DISPLAY\_NONE*** No display.

***DISPLAY\_XV*** Use xv to display.

***DISPLAY\_XWIN*** Use [XWindow](#) to display.

***DISPLAY\_XWINMGD*** Use [XWinManaged](#) to display.

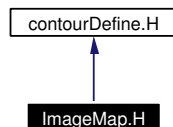
Definition at line 38 of file `ImageDisplayTypes.H`.

Referenced by `convertFromString()`, `ImageDisplay::getType()`, and `ImageDisplay::ImageDisplay()`.

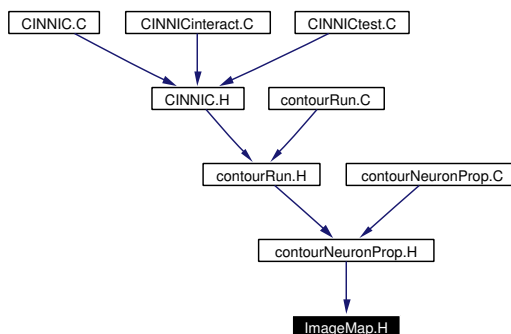
## 11.190 ImageMap.H File Reference

```
#include "contourDefine.H"
```

Include dependency graph for ImageMap.H:



This graph shows which files directly or indirectly include this file:



### Classes

- class [ImageMap](#)  
*3D image map created from the image at diff' orientations*

### 11.190.1 Detailed Description

3D image map created from the image at diff' orientations

Definition in file [ImageMap.H](#).



## 11.191 ImageSet.C File Reference

```
#include "ImageSet.H"
#include "Image.H"
#include "log.H"
#include "Point2D.H"
#include <algorithm>
#include <cmath>
```

Include dependency graph for ImageSet.C:

### Defines

- #define INSTANTIATE(T) template class [ImageSet](#)< T >;

### 11.191.1 Detailed Description

Implementation for image pyramid template class

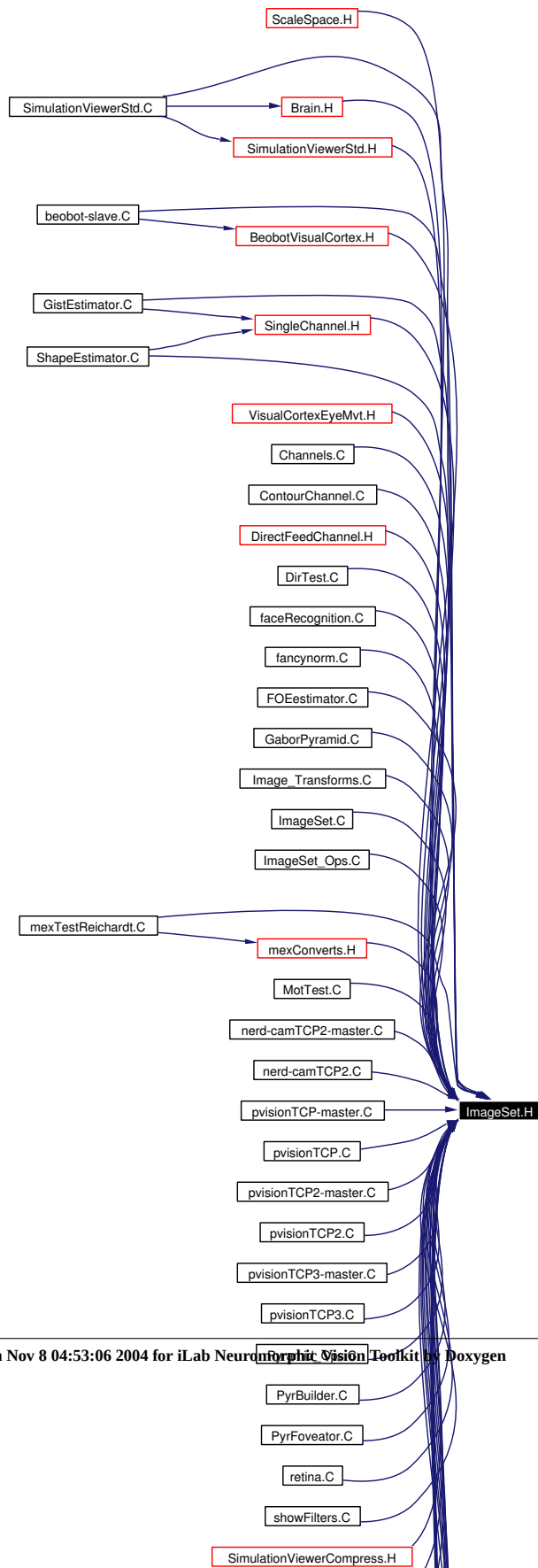
Definition in file [ImageSet.C](#).

## 11.192 ImageSet.H File Reference

```
#include "saliency.H"  
#include "ArrayData.H"  
#include <deque>
```

Include dependency graph for ImageSet.H:

This graph shows which files directly or indirectly include this file:



## Classes

- class [ImageSet](#)

*This class implements a set of images, often used as a dyadic pyramid.*

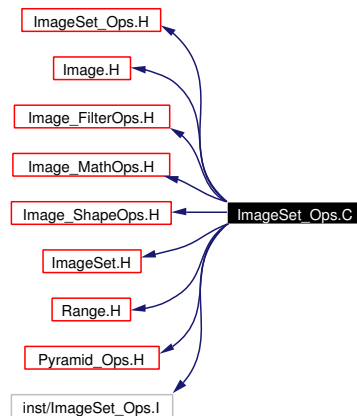
### 11.192.1 Detailed Description

Definition in file [ImageSet.H](#).

## 11.193 ImageSet\_Ops.C File Reference

```
#include "ImageSet_Ops.H"
#include "Image.H"
#include "Image_FilterOps.H"
#include "Image_MathOps.H"
#include "Image_ShapeOps.H"
#include "ImageSet.H"
#include "Range.H"
#include "Pyramid_Ops.H"
#include "inst/ImageSet_Ops.I"
```

Include dependency graph for ImageSet\_Ops.C:



## Functions

- `template<class T> bool isHomogeneous (const ImageSet< T > &x)`  
Returns true if all images in the set are the same size.
- `template<class T> Image< T > sum (const ImageSet< T > &x)`  
Returns the sum of all the images in the set. Requires `isHomogeneous()`.
- `template<class T> Image< T > mean (const ImageSet< T > &x)`  
Returns the sum of all the images in the set. Requires `isHomogeneous()`.
- `template<class T> Range< T > rangeOf (const ImageSet< T > &x)`

*Returns the overall range of all images in the set.*

- `template<class T> ImageSet< T > takeSlice (const ImageSet< T > *sets, uint nsets, uint level)`

*Make an ImageSet from the s'th level from each of the N input sets.*

- `template<class T> Image< T > makeImageArray (const ImageSet< T > &x, int Nx, int grid_width, T grid_color, int destX, int destY)`

*Make a single image-array from the set (see [concatArray\(\)](#) in *Image\_ShapeOps*).*

- `template<class T> ImageSet< T > reduce (const ImageSet< T > &x, int octaves)`

*Make a new ImageSet by downscaling the input by the given number of octaves.*

- `template<class T> ImageSet< T > rescale (const ImageSet< T > &x, const Dims &dims)`

*Make a new ImageSet by rescaling each input image to the given dimensions.*

- `ImageSet< float > orientedFilterSet (const Image< float > &lowPassedInput, float period, const float *angles, const uint numAngles)`

*Generate a set of oriented filter outputs for a given set of orientations.*

- `template<class T> ImageSet< T > & operator-= (ImageSet< T > &x, const Image< T > &y)`

*x -= y, for each image in x; requires isHomogeneous(x).*

- `template<class T> ImageSet< T > & operator+= (ImageSet< T > &x, const Image< T > &y)`

*x += y, for each image in x; requires isHomogeneous(x).*

- `template<class T> ImageSet< T > & operator *= (ImageSet< T > &x, const Image< T > &y)`

*x \*= y, for each image in x; requires isHomogeneous(x).*

- `template<class T> ImageSet< T > & operator/= (ImageSet< T > &x, const Image< T > &y)`

*x /= y, for each image in x; requires isHomogeneous(x).*

- `template<class T> ImageSet< T > & operator-= (ImageSet< T > &x, const T &v)`

*x -= v, for each image in x*

- `template<class T> ImageSet< T > & operator+= (ImageSet< T > &x, const T &v)`

$x += v$ , for each image in  $x$

- `template<class T> ImageSet< T > & operator += (ImageSet< T > &x, const T &v)`  
 $x += v$ , for each image in  $x$
- `template<class T> ImageSet< T > & operator /= (ImageSet< T > &x, const T &v)`  
 $x /= v$ , for each image in  $x$
- `template<class T> ImageSet< T > operator- (ImageSet< T > &x, const T &v)`  
 $x - v$ , for each image in  $x$
- `template<class T> ImageSet< T > operator+ (ImageSet< T > &x, const T &v)`  
 $x + v$ , for each image in  $x$
- `template<class T> ImageSet< T > operator * (ImageSet< T > &x, const T &v)`  
 $x * v$ , for each image in  $x$
- `template<class T> ImageSet< T > operator/ (ImageSet< T > &x, const T &v)`  
 $x / v$ , for each image in  $x$
- `template<class T> ImageSet< T > & operator-= (ImageSet< T > &x, const ImageSet< T > &y)`  
 $x -= y$ , for each image in  $x$  with the respective image in  $y$
- `template<class T> ImageSet< T > & operator+= (ImageSet< T > &x, const ImageSet< T > &y)`  
 $x += y$ , for each image in  $x$  with the respective image in  $y$
- `template<class T> ImageSet< T > & operator *= (ImageSet< T > &x, const ImageSet< T > &y)`  
 $x *= y$ , for each image in  $x$  with the respective image in  $y$
- `template<class T> ImageSet< T > & operator/= (ImageSet< T > &x, const ImageSet< T > &y)`  
 $x /= y$ , for each image in  $x$  with the respective image in  $y$
- `template<class T> ImageSet< T > clampedDiff (const ImageSet< T > &b, const ImageSet< T > &c)`

*clampedDiff(b,c) for each image*

- `template<class T> void doRectify (ImageSet< T > &x)`  
 *$x = \text{rectify}(x)$ , for each image in  $x$*
- `template<class T> void doLowThresh (ImageSet< T > &x, const T threshold, const T newval)`  
*For each image in  $x$ , replace values  $< \text{threshold}$  with  $\text{newval}$ .*
- `template<class T> void doLowThreshAbs (ImageSet< T > &x, const T threshold, const T newval)`  
*For each image in  $x$ , replace values whose abs is  $< \text{threshold}$  with  $\text{newval}$ .*
- `template<class T> void doSqrt (ImageSet< T > &x)`  
*For each image in  $x$ , take the square root at every pixel.*
- `void doMeanNormalize (ImageSet< float > &x)`  
*Subtract the overall mean from each image.*
- `void doOneNormalize (ImageSet< float > &x)`  
*Normalize so overall max value is 1.0.*
- `void doEnergyNorm (ImageSet< float > &x)`  
*Divide image by the local image energy, then subtract overall mean.*
- `void doApplyBiases (ImageSet< float > &x, const float *biases)`  
*Multiply each image by the corresponding bias.*
- `void doAddWeighted (ImageSet< float > &x, const ImageSet< float > &y, float multiple)`  
 *$x += v * y$ , for each image in  $x$  and  $y$ , respectively*
- `void doClear (ImageSet< float > &x, float v)`  
 *$x = v$ , for each image in  $x$*

### 11.193.1 Detailed Description

Free functions operating on sets of images

Definition in file [ImageSet\\_Ops.C](#).



## 11.193.2 Function Documentation

### 11.193.2.1 `template<class T> ImageSet<T> clampedDiff (const ImageSet< T > & b, const ImageSet< T > & c)`

`clampedDiff(b,c)` for each image

Definition at line 392 of file `ImageSet_Ops.C`.

References `ASSERT`, `ImageSet< T >::size()`, and `uint`.

Referenced by `ImageCacheAvg< T >::clampedDiffMean()`, and `Orientation-Channel::doInteractions()`.

### 11.193.2.2 `void doAddWeighted (ImageSet< float > & x, const ImageSet< float > & y, float multiple)`

`x += v * y`, for each image in `x` and `y`, respectively

Definition at line 477 of file `ImageSet_Ops.C`.

References `ASSERT`, `ImageSet< T >::size()`, and `uint`.

### 11.193.2.3 `void doApplyBiases (ImageSet< float > & x, const float * biases)`

Multiply each image by the corresponding bias.

Definition at line 468 of file `ImageSet_Ops.C`.

References `ImageSet< T >::size()`, and `uint`.

### 11.193.2.4 `void doClear (ImageSet< float > & x, float v)`

`x = v`, for each image in `x`

Definition at line 485 of file `ImageSet_Ops.C`.

References `ImageSet< T >::clear()`, `ImageSet< T >::size()`, and `uint`.

### 11.193.2.5 `void doEnergyNorm (ImageSet< float > & x)`

Divide image by the local image energy, then subtract overall mean.

Definition at line 459 of file `ImageSet_Ops.C`.

References `energyNorm()`, `ImageSet< T >::getDepth()`, and `uint`.

**11.193.2.6** `template<class T> void doLowThresh (ImageSet< T > & x, const T threshold, const T newval)` [inline]

For each image in x, replace values < threshold with newval.

Definition at line 419 of file ImageSet\_Ops.C.

References `ImageSet< T >::size()`, and `uint`.

Referenced by `SingleChannel::computePyramid()`, `OrientationChannel::doInteractions()`, and `SingleChannel::setClipPyramid()`.

**11.193.2.7** `template<class T> void doLowThreshAbs (ImageSet< T > & x, const T threshold, const T newval)` [inline]

For each image in x, replace values whose abs is < threshold with newval.

Definition at line 427 of file ImageSet\_Ops.C.

References `ImageSet< T >::size()`, and `uint`.

Referenced by `SingleChannel::computePyramid()`.

**11.193.2.8** `void doMeanNormalize (ImageSet< float > & x)`

Subtract the overall mean from each image.

Definition at line 442 of file ImageSet\_Ops.C.

References `mean()`.

**11.193.2.9** `void doOneNormalize (ImageSet< float > & x)`

Normalize so overall max value is 1.0.

Definition at line 448 of file ImageSet\_Ops.C.

References `Range< T >::max()`, `rangeOf()`, `ImageSet< T >::size()`, and `uint`.

**11.193.2.10** `template<class T> void doRectify (ImageSet< T > & x)`  
[inline]

`x = rectify(x)`, for each image in x

Definition at line 411 of file ImageSet\_Ops.C.

References `ImageSet< T >::size()`, and `uint`.

Referenced by `SingleChannel::computePyramid()`.

**11.193.2.11** `template<class T> void doSqrt (ImageSet< T > & x) [inline]`

For each image in x, take the square root at every pixel.

Definition at line 435 of file ImageSet\_Ops.C.

References ImageSet< T >::size(), sqrt(), and uint.

Referenced by DirectionChannel::doInput().

**11.193.2.12** `template<class T> bool isHomogeneous (const ImageSet< T > & x)`

Returns true if all images in the set are the same size.

Also returns true for the empty set.

Definition at line 54 of file ImageSet\_Ops.C.

References ImageSet< T >::size(), and uint.

Referenced by operator \*(), operator +=(), operator -=(), operator /=(), and sum().

**11.193.2.13** `template<class T> Image<T> makeImageArray (const ImageSet< T > & x, int Nx = -1, int grid_width = 1, T grid_color = T(), int destX = -1, int destY = -1)`

Make a single image-array from the set (see [concatArray\(\)](#) in Image\_ShapeOps).

**Parameters:**

*Nx* Number of columns; if negative, the result will be roughly square.

*grid\_width* Pixel width of grid separators to be overlaid on image.

*grid\_color* Color in which to draw grid.

*destX* Width to which each image is reshaped; if negative, reshape to size of first image.

*destY* Height to which each image is reshaped; if negative, reshape to size of first image.

Definition at line 124 of file ImageSet\_Ops.C.

References [concatArray\(\)](#), Image< T >::drawGrid(), [isDyadic\(\)](#), ImageSet< T >::size(), sqrt(), and [zoomXY\(\)](#).

**11.193.2.14** `template<class T> Image<T> mean (const ImageSet< T > & x)`

Returns the sum of all the images in the set. Requires [isHomogeneous\(\)](#).

Definition at line 85 of file ImageSet\_Ops.C.

References ImageSet< T >::size(), and sum().

Referenced by dogFilterHmax(), doMeanNormalize(), energyNorm(), gaborFilter3(), Hmax::init(), learningCoeff(), and main().

**11.193.2.15** `template<class T> ImageSet<T> operator * (ImageSet< T > & x, const T & v)`

$x * v$ , for each image in  $x$

Definition at line 328 of file ImageSet\_Ops.C.

**11.193.2.16** `template<class T> ImageSet<T>& operator *= (ImageSet< T > & x, const ImageSet< T > & y)`

$x *= y$ , for each image in  $x$  with the respective image in  $y$

Definition at line 368 of file ImageSet\_Ops.C.

References ASSERT, ImageSet< T >::size(), and uint.

**11.193.2.17** `template<class T> ImageSet<T>& operator *= (ImageSet< T > & x, const T & v)`

$x *= v$ , for each image in  $x$

Definition at line 292 of file ImageSet\_Ops.C.

**11.193.2.18** `template<class T> ImageSet<T>& operator *= (ImageSet< T > & x, const Image< T > & y)`

$x *= y$ , for each image in  $x$ ; requires isHomogeneous( $x$ ).

Definition at line 248 of file ImageSet\_Ops.C.

**11.193.2.19** `template<class T> ImageSet<T> operator+ (ImageSet< T > & x, const T & v)`

$x + v$ , for each image in  $x$

Definition at line 320 of file ImageSet\_Ops.C.

**11.193.2.20** `template<class T> ImageSet<T>& operator+= (ImageSet< T > & x, const ImageSet< T > & y)`

$x += y$ , for each image in  $x$  with the respective image in  $y$

Definition at line 356 of file ImageSet\_Ops.C.

References ASSERT, ImageSet< T >::size(), and uint.

**11.193.2.21** `template<class T> ImageSet<T>& operator+= (ImageSet< T > & x, const T & v)`

$x += v$ , for each image in  $x$

Definition at line 282 of file ImageSet\_Ops.C.

**11.193.2.22** `template<class T> ImageSet<T>& operator+= (ImageSet< T > & x, const Image< T > & y)`

$x += y$ , for each image in  $x$ ; requires isHomogeneous( $x$ ).

Definition at line 236 of file ImageSet\_Ops.C.

**11.193.2.23** `template<class T> ImageSet<T> operator- (ImageSet< T > & x, const T & v)`

$x - v$ , for each image in  $x$

Definition at line 312 of file ImageSet\_Ops.C.

**11.193.2.24** `template<class T> ImageSet<T>& operator-= (ImageSet< T > & x, const ImageSet< T > & y)`

$x -= y$ , for each image in  $x$  with the respective image in  $y$

Definition at line 344 of file ImageSet\_Ops.C.

References ASSERT, ImageSet< T >::size(), and uint.

**11.193.2.25** `template<class T> ImageSet<T>& operator-= (ImageSet< T > & x, const T & v)`

$x -= v$ , for each image in  $x$

Definition at line 272 of file ImageSet\_Ops.C.

**11.193.2.26** `template<class T> ImageSet<T>& operator-= (ImageSet< T > & x, const Image< T > & y)`

x -= y, for each image in x; requires isHomogeneous(x).

Definition at line 224 of file `ImageSet_Ops.C`.

**11.193.2.27** `template<class T> ImageSet<T> operator/ (ImageSet< T > & x, const T & v)`

x / v, for each image in x

Definition at line 336 of file `ImageSet_Ops.C`.

**11.193.2.28** `template<class T> ImageSet<T>& operator/= (ImageSet< T > & x, const ImageSet< T > & y)`

x /= y, for each image in x with the respective image in y

Definition at line 380 of file `ImageSet_Ops.C`.

References `ASSERT`, `ImageSet< T >::size()`, and `uint`.

**11.193.2.29** `template<class T> ImageSet<T>& operator/= (ImageSet< T > & x, const T & v)`

x /= v, for each image in x

Definition at line 302 of file `ImageSet_Ops.C`.

**11.193.2.30** `template<class T> ImageSet<T>& operator/= (ImageSet< T > & x, const Image< T > & y)`

x /= y, for each image in x; requires isHomogeneous(x).

Definition at line 260 of file `ImageSet_Ops.C`.

**11.193.2.31** `ImageSet<float> orientedFilterSet (const Image< float > & lowPassedInput, float period, const float * angles, const uint numAngles)`

Generate a set of oriented filter outputs for a given set of orientations.

Caller is responsible for passing input in the form of (image - `lowPass(image)`)

Definition at line 199 of file `ImageSet_Ops.C`.

References `orientedFilter()`, and `uint`.

**11.193.2.32** `template<class T> Range<T> rangeOf (const ImageSet< T > & x)`

Returns the overall range of all images in the set.

Definition at line 94 of file `ImageSet_Ops.C`.

References `Range< T >::merge()`, `ImageSet< T >::size()`, and `uint`.

Referenced by `doOneNormalize()`, and `normalizeWithScale()`.

**11.193.2.33** `template<class T> ImageSet<T> reduce (const ImageSet< T > & x, int octaves)`

Make a new `ImageSet` by downscaling the input by the given number of octaves.

Definition at line 168 of file `ImageSet_Ops.C`.

References `decXY()`, `lowPass3()`, `ImageSet< T >::size()`, and `uint`.

**11.193.2.34** `template<class T> ImageSet<T> rescale (const ImageSet< T > & x, const Dims & dims)`

Make a new `ImageSet` by rescaling each input image to the given dimensions.

Definition at line 186 of file `ImageSet_Ops.C`.

References `ImageSet< T >::size()`, and `uint`.

Referenced by `SimulationViewerEyeMvt2::attentionShift()`, `SoxChannel::combineOutputs()`, `DirectFeedChannel::computeOutput()`, `InferoTemporal::computeVisualObject()`, `MbariResultViewer::displayImage()`, `downSizeClean()`, `SimulationViewerStd::drawMegaCombo()`, `contourRun::dumpEnergySigmoid()`, `TemplateMatchChannel::findBestMatch()`, `PyrFoveator::foveate()`, `SimulationViewerCompress::getTraj()`, `SimulationViewerEyeMvt::getTraj()`, `SimulationViewerEyeMvt2::getTraj()`, `SimulationViewerStd::getTraj()`, `TaskRelevanceMapStd::input()`, `VisualBuffer::input()`, `SingleChannel::learn()`, `main()`, `maxNormalizeFancy()`, `maxNormalizeFancyFast()`, `SingleChannel::postProcessMap()`, `FrameSeries::resizeImage()`, and `TaskRelevanceMapStd::update()`.

**11.193.2.35** `template<class T> Image<T> sum (const ImageSet< T > & x)`

Returns the sum of all the images in the set. Requires `isHomogeneous()`.

Definition at line 69 of file `ImageSet_Ops.C`.

References ASSERT, isHomogeneous(), ImageSet< T >::size(), uint, and ZEROS.

Referenced by BitObject::doesIntersect(), dogFilterHmax(), ShapeEstimator::getObjectArea(), Hmax::init(), learningCoeff(), main(), and mean().

**11.193.2.36** `template<class T> ImageSet<T> takeSlice (const ImageSet< T >  
* sets, uint N, uint s)`

Make an ImageSet from the s'th level from each of the N input sets.

This is useful, for example, in transforming an array of ImageSets from multiple-scales-per-set/one-orientation-per-set into the reverse, multiple-orientations-per-set/one-scale-per-set.

Definition at line 108 of file ImageSet\_Ops.C.

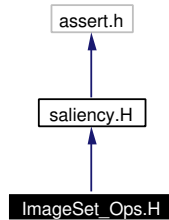
References uint.



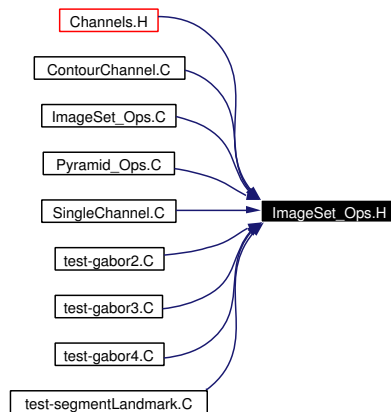
## 11.194 ImageSet\_Ops.H File Reference

```
#include "saliency.H"
```

Include dependency graph for ImageSet\_Ops.H:



This graph shows which files directly or indirectly include this file:



### ImageSet processing functions

Not all of these functions are templates, simply because alternate versions aren't currently needed.

- `template<class T> bool isHomogeneous (const ImageSet< T > &x)`  
Returns true if all images in the set are the same size.
- `template<class T> Image< T > sum (const ImageSet< T > &x)`  
Returns the sum of all the images in the set. Requires `isHomogeneous()`.

- `template<class T> Image< T > mean (const ImageSet< T > &x)`  
Returns the sum of all the images in the set. Requires *isHomogeneous()*.
- `template<class T> Range< T > rangeOf (const ImageSet< T > &x)`  
Returns the overall range of all images in the set.
- `template<class T> ImageSet< T > takeSlice (const ImageSet< T > *sets, uint N, uint s)`  
Make an *ImageSet* from the *s*'th level from each of the *N* input sets.
- `template<class T> Image< T > makeImageArray (const ImageSet< T > &x, int Nx=-1, int grid_width=1, T grid_color=T(), int destX=-1, int destY=-1)`  
Make a single image-array from the set (see *concatArray()* in *Image\_ShapeOps*).
- `template<class T> ImageSet< T > reduce (const ImageSet< T > &x, int octaves)`  
Make a new *ImageSet* by downscaling the input by the given number of octaves.
- `template<class T> ImageSet< T > rescale (const ImageSet< T > &x, const Dims &dims)`  
Make a new *ImageSet* by rescaling each input image to the given dimensions.
- `ImageSet< float > orientedFilterSet (const Image< float > &lowPassedInput, float period, const float *angles, const uint numAngles)`  
Generate a set of oriented filter outputs for a given set of orientations.

## ImageSet mathematical operators

This set of operators is obviously incomplete; more operators can be added as they are needed. There are three basic kinds of operators:

- (1) ImageSet-scalar ops. These place no restrictions on the *ImageSet*.
- (2) ImageSet-Image ops. These require the *ImageSet* to be homogeneous (all images the same size), and require the *Image* argument to be the same size as well.
- (3) ImageSet-ImageSet ops. These require the two sets to have the same number of images, and for each pair of corresponding images to be the same size.

- `template<class T> ImageSet< T > & operator-= (ImageSet< T > &x, const Image< T > &y)`  
*x -= y, for each image in x; requires isHomogeneous(x).*

- `template<class T> ImageSet< T > & operator+= (ImageSet< T > &x, const Image< T > &y)`  
 $x += y$ , for each image in  $x$ ; requires `isHomogeneous(x)`.
- `template<class T> ImageSet< T > & operator *= (ImageSet< T > &x, const Image< T > &y)`  
 $x *= y$ , for each image in  $x$ ; requires `isHomogeneous(x)`.
- `template<class T> ImageSet< T > & operator/= (ImageSet< T > &x, const Image< T > &y)`  
 $x /= y$ , for each image in  $x$ ; requires `isHomogeneous(x)`.
- `template<class T> ImageSet< T > & operator-= (ImageSet< T > &x, const T &v)`  
 $x -= v$ , for each image in  $x$
- `template<class T> ImageSet< T > & operator+= (ImageSet< T > &x, const T &v)`  
 $x += v$ , for each image in  $x$
- `template<class T> ImageSet< T > & operator *= (ImageSet< T > &x, const T &v)`  
 $x *= v$ , for each image in  $x$
- `template<class T> ImageSet< T > & operator/= (ImageSet< T > &x, const T &v)`  
 $x /= v$ , for each image in  $x$
- `template<class T> ImageSet< T > operator- (ImageSet< T > &x, const T &v)`  
 $x - v$ , for each image in  $x$
- `template<class T> ImageSet< T > operator+ (ImageSet< T > &x, const T &v)`  
 $x + v$ , for each image in  $x$
- `template<class T> ImageSet< T > operator * (ImageSet< T > &x, const T &v)`  
 $x * v$ , for each image in  $x$
- `template<class T> ImageSet< T > operator/ (ImageSet< T > &x, const T &v)`  
 $x / v$ , for each image in  $x$

- `template<class T> ImageSet< T > & operator-= (ImageSet< T > &x, const ImageSet< T > &y)`  
 *$x -= y$ , for each image in  $x$  with the respective image in  $y$*
- `template<class T> ImageSet< T > & operator+= (ImageSet< T > &x, const ImageSet< T > &y)`  
 *$x += y$ , for each image in  $x$  with the respective image in  $y$*
- `template<class T> ImageSet< T > & operator *= (ImageSet< T > &x, const ImageSet< T > &y)`  
 *$x *= y$ , for each image in  $x$  with the respective image in  $y$*
- `template<class T> ImageSet< T > & operator/= (ImageSet< T > &x, const ImageSet< T > &y)`  
 *$x /= y$ , for each image in  $x$  with the respective image in  $y$*
- `template<class T> ImageSet< T > clampedDiff (const ImageSet< T > &b, const ImageSet< T > &c)`  
 *$clampedDiff(b,c)$  for each image*

## In-place ImageSet modification functions

Any functions which modify an `ImageSet` in-place should adhere to the following conventions:

- (1) The to-be-modified `ImageSet` should be passed by non-const reference as the first argument to the function.
- (2) The function should be named with a "do" prefix, to distinguish it from functions that work by returning a new result instead of modifying an argument in place.

Not all of these functions are templates, simply because alternate versions aren't currently needed.

- `template<class T> void doRectify (ImageSet< T > &x)`  
 *$x = rectify(x)$ , for each image in  $x$*
- `template<class T> void doLowThresh (ImageSet< T > &x, const T threshold, const T newval=T())`  
*For each image in  $x$ , replace values  $< threshold$  with  $newval$ .*
- `template<class T> void doLowThreshAbs (ImageSet< T > &x, const T threshold, const T newval=T())`

*For each image in x, replace values whose abs is < threshold with newval.*

- `template<class T> void doSqrt (ImageSet< T > &x)`  
*For each image in x, take the square root at every pixel.*
- `void doMeanNormalize (ImageSet< float > &x)`  
*Subtract the overall mean from each image.*
- `void doOneNormalize (ImageSet< float > &x)`  
*Normalize so overall max value is 1.0.*
- `void doEnergyNorm (ImageSet< float > &x)`  
*Divide image by the local image energy, then subtract overall mean.*
- `void doApplyBiases (ImageSet< float > &x, const float *biases)`  
*Multiply each image by the corresponding bias.*
- `void doAddWeighted (ImageSet< float > &x, const ImageSet< float > &y, float v)`  
 *$x += v * y$ , for each image in x and y, respectively*
- `void doClear (ImageSet< float > &x, float v)`  
 *$x = v$ , for each image in x*

### 11.194.1 Detailed Description

Free functions operating on sets of images

Definition in file [ImageSet\\_Ops.H](#).

### 11.194.2 Function Documentation

#### 11.194.2.1 `template<class T> ImageSet<T> clampedDiff (const ImageSet< T > &b, const ImageSet< T > &c)`

`clampedDiff(b,c)` for each image

Definition at line 392 of file [ImageSet\\_Ops.C](#).

References `ASSERT`, `ImageSet< T >::size()`, and `uint`.

Referenced by `ImageCacheAvg< T >::clampedDiffMean()`, and `Orientation-Channel::doInteractions()`.

**11.194.2.2 void doAddWeighted (ImageSet< float > & x, const ImageSet< float > & y, float v)**

$x += v * y$ , for each image in x and y, respectively

Definition at line 477 of file ImageSet\_Ops.C.

References ASSERT, ImageSet< T >::size(), and uint.

**11.194.2.3 void doApplyBiases (ImageSet< float > & x, const float \* biases)**

Multiply each image by the corresponding bias.

Definition at line 468 of file ImageSet\_Ops.C.

References ImageSet< T >::size(), and uint.

**11.194.2.4 void doClear (ImageSet< float > & x, float v)**

$x = v$ , for each image in x

Definition at line 485 of file ImageSet\_Ops.C.

References ImageSet< T >::clear(), ImageSet< T >::size(), and uint.

**11.194.2.5 void doEnergyNorm (ImageSet< float > & x)**

Divide image by the local image energy, then subtract overall mean.

Definition at line 459 of file ImageSet\_Ops.C.

References energyNorm(), ImageSet< T >::getDepth(), and uint.

**11.194.2.6 template<class T> void doLowThresh (ImageSet< T > & x, const T threshold, const T newval = T()) [inline]**

For each image in x, replace values < threshold with newval.

Definition at line 419 of file ImageSet\_Ops.C.

References ImageSet< T >::size(), and uint.

Referenced by SingleChannel::computePyramid(), OrientationChannel::do-Interactions(), and SingleChannel::setClipPyramid().

**11.194.2.7** `template<class T> void doLowThreshAbs (ImageSet< T > & x,  
const T threshold, const T newval = T()) [inline]`

For each image in x, replace values whose abs is < threshold with newval.

Definition at line 427 of file ImageSet\_Ops.C.

References ImageSet< T >::size(), and uint.

Referenced by SingleChannel::computePyramid().

**11.194.2.8** `void doMeanNormalize (ImageSet< float > & x)`

Subtract the overall mean from each image.

Definition at line 442 of file ImageSet\_Ops.C.

References mean().

**11.194.2.9** `void doOneNormalize (ImageSet< float > & x)`

Normalize so overall max value is 1.0.

Definition at line 448 of file ImageSet\_Ops.C.

References Range< T >::max(), rangeOf(), ImageSet< T >::size(), and uint.

**11.194.2.10** `template<class T> void doRectify (ImageSet< T > & x)  
[inline]`

$x = \text{rectify}(x)$ , for each image in x

Definition at line 411 of file ImageSet\_Ops.C.

References ImageSet< T >::size(), and uint.

Referenced by SingleChannel::computePyramid().

**11.194.2.11** `template<class T> void doSqrt (ImageSet< T > & x) [inline]`

For each image in x, take the square root at every pixel.

Definition at line 435 of file ImageSet\_Ops.C.

References ImageSet< T >::size(), sqrt(), and uint.

Referenced by DirectionChannel::doInput().

#### 11.194.2.12 `template<class T> bool isHomogeneous (const ImageSet< T > & x)`

Returns true if all images in the set are the same size.

Also returns true for the empty set.

Definition at line 54 of file `ImageSet_Ops.C`.

References `ImageSet< T >::size()`, and `uint`.

Referenced by operator `*=()`, operator `+=()`, operator `-=()`, operator `/=()`, and `sum()`.

#### 11.194.2.13 `template<class T> Image<T> makeImageArray (const ImageSet< T > & x, int Nx = -1, int grid_width = 1, T grid_color = T(), int destX = -1, int destY = -1)`

Make a single image-array from the set (see [concatArray\(\)](#) in `Image_ShapeOps`).

##### Parameters:

**Nx** Number of columns; if negative, the result will be roughly square.

**grid\_width** Pixel width of grid separators to be overlaid on image.

**grid\_color** Color in which to draw grid.

**destX** Width to which each image is reshaped; if negative, reshape to size of first image.

**destY** Height to which each image is reshaped; if negative, reshape to size of first image.

Definition at line 124 of file `ImageSet_Ops.C`.

References `concatArray()`, `Image< T >::drawGrid()`, `isDyadic()`, `ImageSet< T >::size()`, `sqrt()`, and `zoomXY()`.

#### 11.194.2.14 `template<class T> Image<T> mean (const ImageSet< T > & x)`

Returns the sum of all the images in the set. Requires [isHomogeneous\(\)](#).

Definition at line 85 of file `ImageSet_Ops.C`.

References `ImageSet< T >::size()`, and `sum()`.

Referenced by `dogFilterHmax()`, `doMeanNormalize()`, `energyNorm()`, `gaborFilter3()`, `Hmax::init()`, `learningCoeff()`, and `main()`.



**11.194.2.15** `template<class T> ImageSet<T> operator * (ImageSet< T > & x, const T & v)`

$x * v$ , for each image in  $x$

Definition at line 328 of file ImageSet\_Ops.C.

**11.194.2.16** `template<class T> ImageSet<T>& operator *= (ImageSet< T > & x, const ImageSet< T > & y)`

$x *= y$ , for each image in  $x$  with the respective image in  $y$

Definition at line 368 of file ImageSet\_Ops.C.

References ASSERT, ImageSet< T >::size(), and uint.

**11.194.2.17** `template<class T> ImageSet<T>& operator *= (ImageSet< T > & x, const T & v)`

$x *= v$ , for each image in  $x$

Definition at line 292 of file ImageSet\_Ops.C.

References ImageSet< T >::size(), and uint.

**11.194.2.18** `template<class T> ImageSet<T>& operator *= (ImageSet< T > & x, const Image< T > & y)`

$x *= y$ , for each image in  $x$ ; requires isHomogeneous( $x$ ).

Definition at line 248 of file ImageSet\_Ops.C.

References ASSERT, isHomogeneous(), ImageSet< T >::size(), and uint.

**11.194.2.19** `template<class T> ImageSet<T> operator+ (ImageSet< T > & x, const T & v)`

$x + v$ , for each image in  $x$

Definition at line 320 of file ImageSet\_Ops.C.

**11.194.2.20** `template<class T> ImageSet<T>& operator+= (ImageSet< T > & x, const ImageSet< T > & y)`

$x += y$ , for each image in  $x$  with the respective image in  $y$

Definition at line 356 of file ImageSet\_Ops.C.

References ASSERT, ImageSet< T >::size(), and uint.

**11.194.2.21** `template<class T> ImageSet<T>& operator+= (ImageSet< T >  
& x, const T & v)`

$x += v$ , for each image in  $x$

Definition at line 282 of file ImageSet\_Ops.C.

References ImageSet< T >::size(), and uint.

**11.194.2.22** `template<class T> ImageSet<T>& operator+= (ImageSet< T >  
& x, const Image< T > & y)`

$x += y$ , for each image in  $x$ ; requires isHomogeneous( $x$ ).

Definition at line 236 of file ImageSet\_Ops.C.

References ASSERT, isHomogeneous(), ImageSet< T >::size(), and uint.

**11.194.2.23** `template<class T> ImageSet<T> operator- (ImageSet< T > & x,  
const T & v)`

$x - v$ , for each image in  $x$

Definition at line 312 of file ImageSet\_Ops.C.

**11.194.2.24** `template<class T> ImageSet<T>& operator-= (ImageSet< T > &  
x, const ImageSet< T > & y)`

$x -= y$ , for each image in  $x$  with the respective image in  $y$

Definition at line 344 of file ImageSet\_Ops.C.

References ASSERT, ImageSet< T >::size(), and uint.

**11.194.2.25** `template<class T> ImageSet<T>& operator-= (ImageSet< T > &  
x, const T & v)`

$x -= v$ , for each image in  $x$

Definition at line 272 of file ImageSet\_Ops.C.

References ImageSet< T >::size(), and uint.

**11.194.2.26** `template<class T> ImageSet<T>& operator-= (ImageSet< T > & x, const Image< T > & y)`

$x -= y$ , for each image in  $x$ ; requires `isHomogeneous(x)`.

Definition at line 224 of file `ImageSet_Ops.C`.

References `ASSERT`, `isHomogeneous()`, `ImageSet< T >::size()`, and `uint`.

**11.194.2.27** `template<class T> ImageSet<T> operator/ (ImageSet< T > & x, const T & v)`

$x / v$ , for each image in  $x$

Definition at line 336 of file `ImageSet_Ops.C`.

**11.194.2.28** `template<class T> ImageSet<T>& operator/= (ImageSet< T > & x, const ImageSet< T > & y)`

$x /= y$ , for each image in  $x$  with the respective image in  $y$

Definition at line 380 of file `ImageSet_Ops.C`.

References `ASSERT`, `ImageSet< T >::size()`, and `uint`.

**11.194.2.29** `template<class T> ImageSet<T>& operator/= (ImageSet< T > & x, const T & v)`

$x /= v$ , for each image in  $x$

Definition at line 302 of file `ImageSet_Ops.C`.

References `ImageSet< T >::size()`, and `uint`.

**11.194.2.30** `template<class T> ImageSet<T>& operator/= (ImageSet< T > & x, const Image< T > & y)`

$x /= y$ , for each image in  $x$ ; requires `isHomogeneous(x)`.

Definition at line 260 of file `ImageSet_Ops.C`.

References `ASSERT`, `isHomogeneous()`, `ImageSet< T >::size()`, and `uint`.

### 11.194.2.31 **ImageSet**<float> **orientedFilterSet** (const **Image**< float > & *lowPassedInput*, float *period*, const float \* *angles*, const uint *numAngles*)

Generate a set of oriented filter outputs for a given set of orientations.

Caller is responsible for passing input in the form of (image - lowPass(image))

Definition at line 199 of file ImageSet\_Ops.C.

References **orientedFilter()**, and **uint**.

### 11.194.2.32 **template**<class T> **Range**<T> **rangeOf** (const **ImageSet**< T > & *x*)

Returns the overall range of all images in the set.

Definition at line 94 of file ImageSet\_Ops.C.

References **Range**< T >::merge(), **ImageSet**< T >::size(), and **uint**.

Referenced by **doOneNormalize()**, and **normalizeWithScale()**.

### 11.194.2.33 **template**<class T> **ImageSet**<T> **reduce** (const **ImageSet**< T > & *x*, int *octaves*)

Make a new **ImageSet** by downscaling the input by the given number of octaves.

Definition at line 168 of file ImageSet\_Ops.C.

References **decXY()**, **lowPass3()**, **ImageSet**< T >::size(), and **uint**.

### 11.194.2.34 **template**<class T> **ImageSet**<T> **rescale** (const **ImageSet**< T > & *x*, const **Dims** & *dims*)

Make a new **ImageSet** by rescaling each input image to the given dimensions.

Definition at line 186 of file ImageSet\_Ops.C.

References **ImageSet**< T >::size(), and **uint**.

Referenced by **SimulationViewerEyeMvt2::attentionShift()**, **SoxChannel::combineOutputs()**, **DirectFeedChannel::computeOutput()**, **InferoTemporal::computeVisualObject()**, **MbariResultViewer::displayImage()**, **downSizeClean()**, **SimulationViewerStd::drawMegaCombo()**, **contourRun::dumpEnergySigmoid()**, **TemplateMatchChannel::findBestMatch()**, **PyrFoveator::foveate()**, **SimulationViewerStd::getTraj()**, **SimulationViewerEyeMvt2::getTraj()**, **SimulationViewerEyeMvt::getTraj()**, **SimulationViewerCompress::getTraj()**, **VisualBuffer::input()**, **TaskRelevanceMapStd::input()**, **SingleChannel::learn()**, **main()**, **maxNormalizeFancy()**, **maxNormalize-**

FancyFast(), SingleChannel::postProcessMap(), FrameSeries::resizeImage(), and TaskRelevanceMapStd::update().

#### 11.194.2.35 `template<class T> ImageSet<T> sum (const ImageSet< T > & x)`

Returns the sum of all the images in the set. Requires `isHomogeneous()`.

Definition at line 69 of file ImageSet\_Ops.C.

References ASSERT, `isHomogeneous()`, `ImageSet< T >::size()`, `uint`, and ZEROS.

Referenced by `BitObject::doesIntersect()`, `dogFilterHmax()`, `ShapeEstimator::getObjectArea()`, `Hmax::init()`, `learningCoeff()`, `main()`, and `mean()`.

#### 11.194.2.36 `template<class T> ImageSet<T> takeSlice (const ImageSet< T > * sets, uint N, uint s)`

Make an `ImageSet` from the s'th level from each of the N input sets.

This is useful, for example, in transforming an array of `ImageSets` from multiple-scales-per-set/one-orientation-per-set into the reverse, multiple-orientations-per-set/one-scale-per-set.

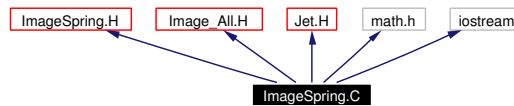
Definition at line 108 of file ImageSet\_Ops.C.

References `uint`.

## 11.195 ImageSpring.C File Reference

```
#include "ImageSpring.H"  
#include "Image_All.H"  
#include "Jet.H"  
#include <math.h>  
#include <iostream>
```

Include dependency graph for ImageSpring.C:



### Defines

- #define **IMAGE\_STRING**
- #define **LIKE\_CENTER** 1.0
- #define **LIKE\_PREVIOUS** 1.0
- #define **LIKE\_BIG** 1.0

### 11.195.1 Detailed Description

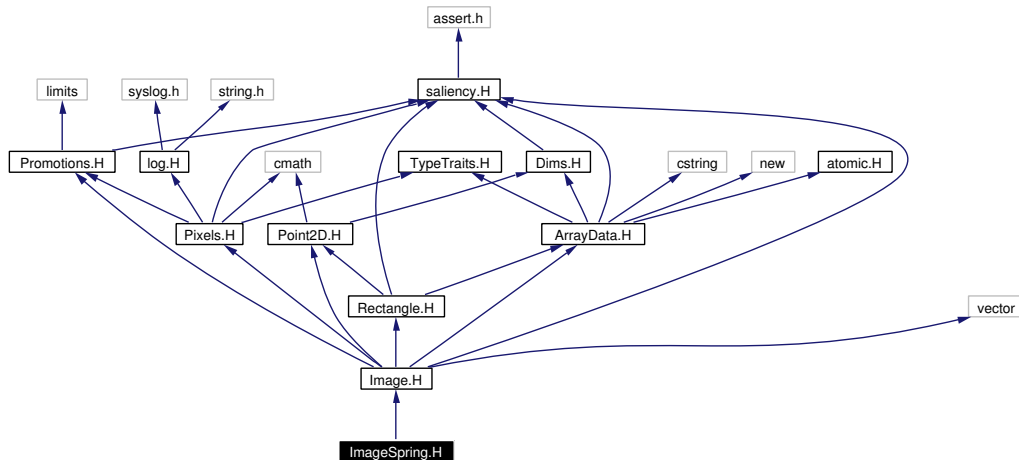
derived from the image template class; all the pixels are linked to theirs neighbors with springs

Definition in file [ImageSpring.C](#).

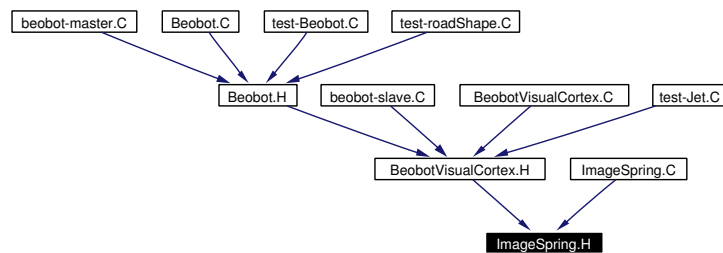
## 11.196 ImageSpring.H File Reference

```
#include "Image.H"
```

Include dependency graph for ImageSpring.H:



This graph shows which files directly or indirectly include this file:



## Classes

- class [ImageSpring](#)

*This class is some sort of self organizing map used for clustering.*

### 11.196.1 Detailed Description

derived from the image template class

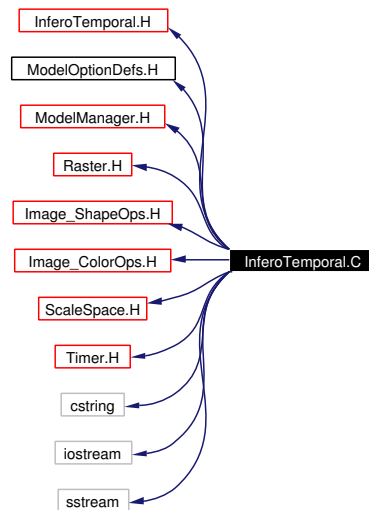
Definition in file [ImageSpring.H](#).



## 11.197 InferoTemporal.C File Reference

```
#include "InferoTemporal.H"  
#include "ModelOptionDefs.H"  
#include "ModelManager.H"  
#include "Raster.H"  
#include "Image_ShapeOps.H"  
#include "Image_ColorOps.H"  
#include "ScaleSpace.H"  
#include "Timer.H"  
#include <cstring>  
#include <iostream>  
#include <sstream>
```

Include dependency graph for InferoTemporal.C:



### 11.197.1 Detailed Description

Object recognition module

Definition in file [InferoTemporal.C](#).

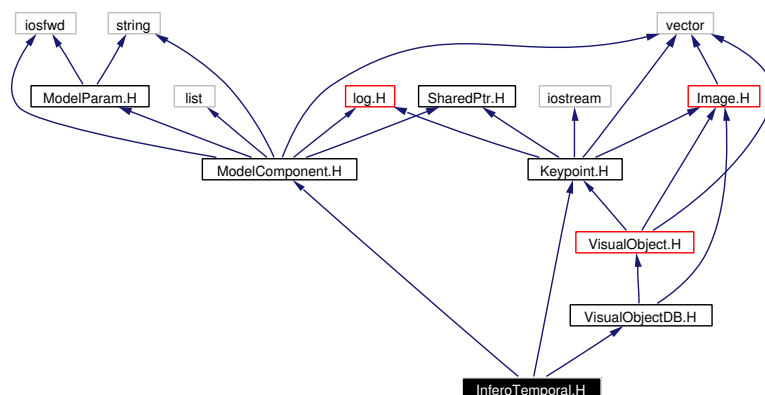
## 11.198 InferoTemporal.H File Reference

```
#include "ModelComponent.H"
```

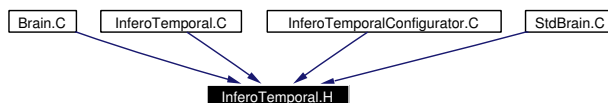
```
#include "Keypoint.H"
```

```
#include "VisualObjectDB.H"
```

Include dependency graph for InferoTemporal.H:



This graph shows which files directly or indirectly include this file:



### Classes

- class [InferoTemporal](#)  
*InferoTemporal.*

### 11.198.1 Detailed Description

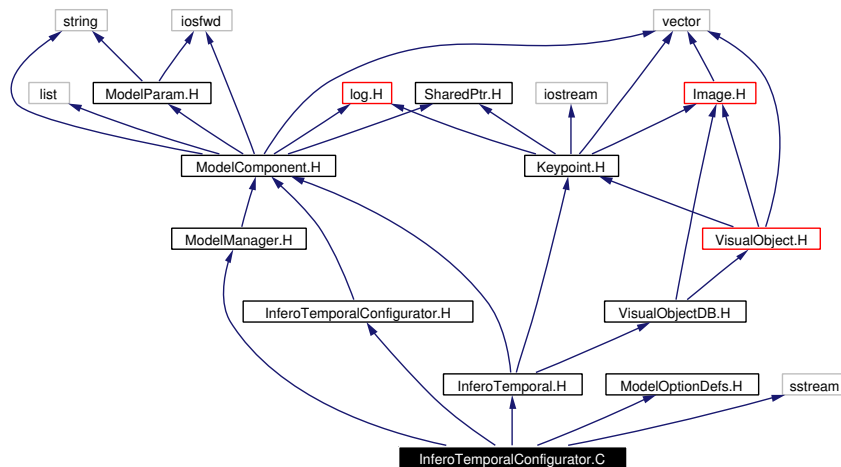
Object recognition module

Definition in file [InferoTemporal.H](#).

## 11.199 InferoTemporalConfigurator.C File Reference

```
#include "InferoTemporalConfigurator.H"
#include "InferoTemporal.H"
#include "ModelManager.H"
#include "ModelOptionDefs.H"
#include <sstream>
```

Include dependency graph for InferoTemporalConfigurator.C:



### 11.199.1 Detailed Description

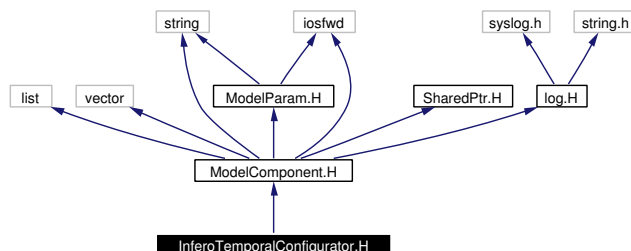
Class to select a [InferoTemporal](#) at runtime

Definition in file [InferoTemporalConfigurator.C](#).

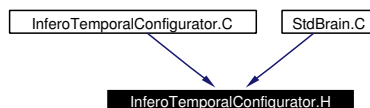
## 11.200 InferoTemporalConfigurator.H File Reference

```
#include "ModelComponent.H"
```

Include dependency graph for InferoTemporalConfigurator.H:



This graph shows which files directly or indirectly include this file:



### Classes

- class [InferoTemporalConfigurator](#)  
*InferoTemporal* configurator.

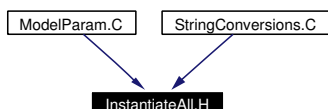
### 11.200.1 Detailed Description

run-time selects which VisualCoxter to use

Definition in file [InferoTemporalConfigurator.H](#).

## 11.201 InstantiateAll.H File Reference

This graph shows which files directly or indirectly include this file:



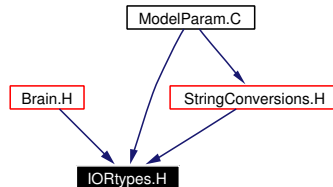
### 11.201.1 Detailed Description

- executes the macro INSTANTIATE for each type for which ModelParam<T> is defined. Typical use of this file: `#define INSTANTIATE(T) template void myFunction1(const T& val); \ template T myFunction2(Image<T>& img, int val); \ template class myClass< T >; #include "InstantiateAll.H"` see [ModelParam.C](#) and [StringConversions.C](#) for example. Extend the list of instantiations if you create a new kind of [ModelParam](#).

Definition in file [InstantiateAll.H](#).

## 11.202 IORtypes.H File Reference

This graph shows which files directly or indirectly include this file:



### Defines

- `#define NBIORTYPES 6`  
*number of known PyramidTypes:*

### Enumerations

- `enum IORtype {`  
`IORnone = 0, IORauto = 1, IORdisc = 2, IORshapeEst = 3,`  
`IORshapeEstSM = 4, IORshapeEstCM = 5 }`  
*Possible types of pyramids.*

### Functions

- `const char * iorTypeName (const IORtype p)`

#### 11.202.1 Detailed Description

Definition of the inhibitionof-return (IOR) types

Definition in file [IORtypes.H](#).

#### 11.202.2 Define Documentation

##### 11.202.2.1 `#define NBIORTYPES 6`

number of known PyramidTypes:

Definition at line 53 of file IORtypes.H.

Referenced by `convertFromString()`.

### 11.202.3 Enumeration Type Documentation

#### 11.202.3.1 enum **IORtype**

Possible types of pyramids.

**Enumeration values:**

**IORnone** No inhibition of return at all.

**IORauto** Determine from [SimulationViewer](#).

**IORdisc** A disk of the radius of the focus of attention.

**IORshapeEst** ShapeEstimator-shaped.

**IORshapeEstSM** ShapeEstimator-shaped, based on saliency map.

**IORshapeEstCM** ShapeEstimator-shaped, based on conspicuity map.

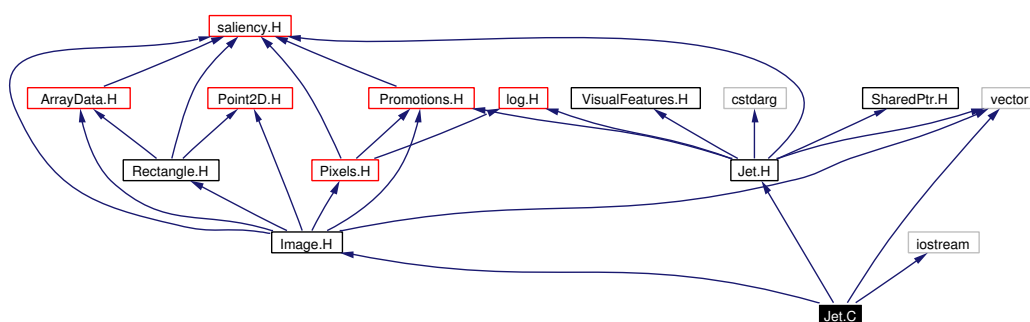
Definition at line 42 of file IORtypes.H.

Referenced by `Brain::Brain()`, and `convertFromString()`.

## 11.203 Jet.C File Reference

```
#include "Jet.H"
#include "Image.H"
#include <vector>
#include <iostream>
```

Include dependency graph for Jet.C:



## Functions

- double [raodistance](#) (const [Jet](#)< float > &j1, const [Jet](#)< float > &j2, const int idxmin, const int idxmax)

*Compute distance between two Jets like Rao et al., Vis Res, 2002.*

### 11.203.1 Detailed Description

a simple jet

Definition in file [Jet.C](#).

### 11.203.2 Function Documentation

#### 11.203.2.1 double raodistance (const [Jet](#)< float > &j1, const [Jet](#)< float > &j2, const int idxmin, const int idxmax)

Compute distance between two Jets like Rao et al., Vis Res, 2002.

Definition at line 249 of file Jet.C.



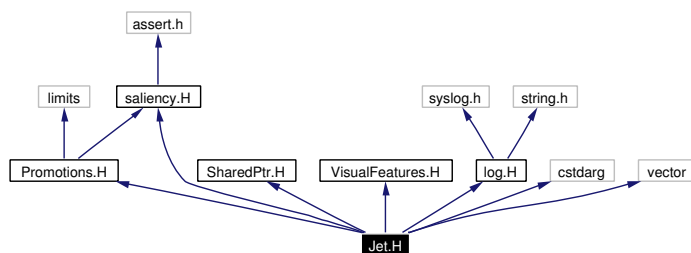
References ASSERT, Jet< T >::getIndexRange(), Jet< T >::getNbIndexRanges(), Jet< T >::getVal(), Jet< T >::hasFeatureType(), NBVISUALFEATURES, RAW, and VisualFeature.

Referenced by VisualCortex::getRaoJetMap().

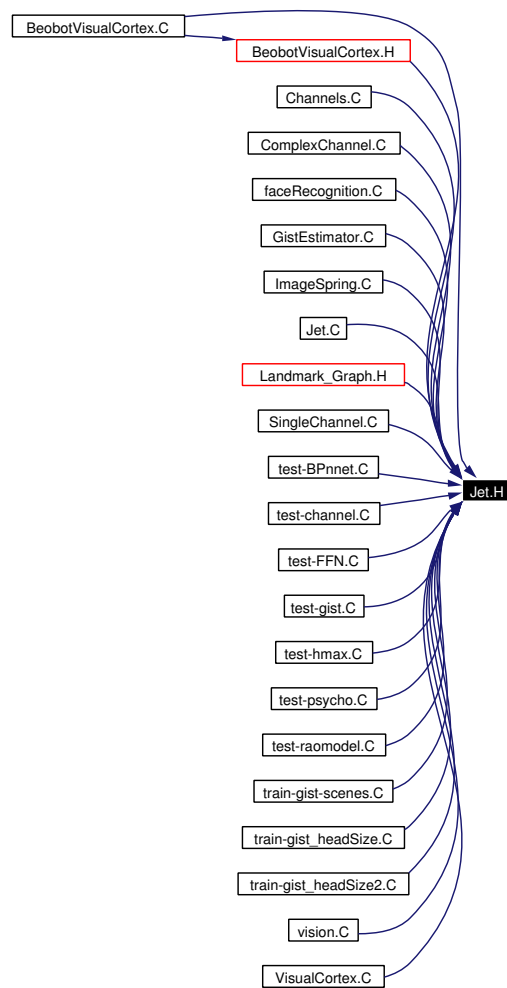
## 11.204 Jet.H File Reference

```
#include "saliency.H"  
#include "SharedPtr.H"  
#include "VisualFeatures.H"  
#include "Promotions.H"  
#include "log.H"  
#include <cstdarg>  
#include <vector>
```

Include dependency graph for Jet.H:



This graph shows which files directly or indirectly include this file:



## Classes

- class [JetSpec](#)  
*This class specifies the features, scales and indices represented in a [Jet](#).*
- struct **JetSpec::JFT**  
*Chained list element that holds index range defs for given feature type.*
- struct **JetSpec::JF**  
*chained list element that holds JTF definitions for given feature*

- class [Jet](#)

*This class specifies a Jet, i.e., array of feature values for one location.*

## Functions

- double [raodistance](#) (const [Jet](#)< float > &j1, const [Jet](#)< float > &j2, const int idxmin, const int idxmax)

*Compute distance between two Jets like Rao et al., Vis Res, 2002.*

- template<class Stream, class T> Stream & **operator**<< (Stream &s, const [Jet](#)< T > &j)
- template<class Stream, class T> Stream & **operator**>> (Stream &s, [Jet](#)< T > &j)

### 11.204.1 Detailed Description

a simple jet (i.e., cortical hypercolumn)

Definition in file [Jet.H](#).

### 11.204.2 Function Documentation

#### 11.204.2.1 double [raodistance](#) (const [Jet](#)< float > &j1, const [Jet](#)< float > &j2, const int idxmin, const int idxmax)

Compute distance between two Jets like Rao et al., Vis Res, 2002.

Definition at line 249 of file Jet.C.

References ASSERT, [Jet](#)< T >::getIndexRange(), [Jet](#)< T >::getNbIndexRanges(), [Jet](#)< T >::getVal(), [Jet](#)< T >::hasFeatureType(), NBVISUALFEATURES, RAW, and VisualFeature.

Referenced by [VisualCortex::getRaoJetMap\(\)](#).

## 11.205 KalmanFilter.C File Reference

```
#include "KalmanFilter.H"  
#include "Image_MathOps.H"  
#include "Image_IO.H"  
#include "log.H"  
#include <cmath>  
#include <iostream>
```

Include dependency graph for KalmanFilter.C:

### Functions

- float **sqr** (float x)

### 11.205.1 Detailed Description

implementation of a 2nd order linear Kalman Filter

Definition in file [KalmanFilter.C](#).

## 11.206 KalmanFilter.H File Reference

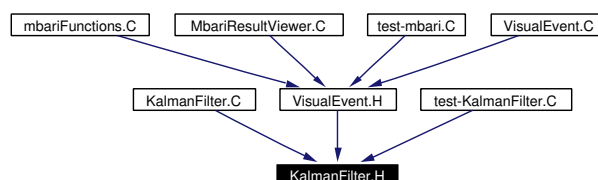
```
#include "saliency.H"
```

```
#include "Image.H"
```

```
#include <fstream>
```

Include dependency graph for KalmanFilter.H:

This graph shows which files directly or indirectly include this file:



### Classes

- class [KalmanFilter](#)  
*implementation of a 2nd order linear Kalman Filter*

### Defines

- #define **DEF\_PNOISE** 1.0F
- #define **DEF\_MNOISE** 1.0F
- #define **DEF\_TSTEP** 1.0F

### 11.206.1 Detailed Description

implementation of a 2nd order linear Kalman Filter

Definition in file [KalmanFilter.H](#).

## 11.207 KDTree.C File Reference

```
#include "KDTree.H"
```

```
#include "log.H"
```

```
#include <math.h>
```

Include dependency graph for KDTree.C:

### 11.207.1 Detailed Description

k-d tree implementation

Definition in file [KDTree.C](#).

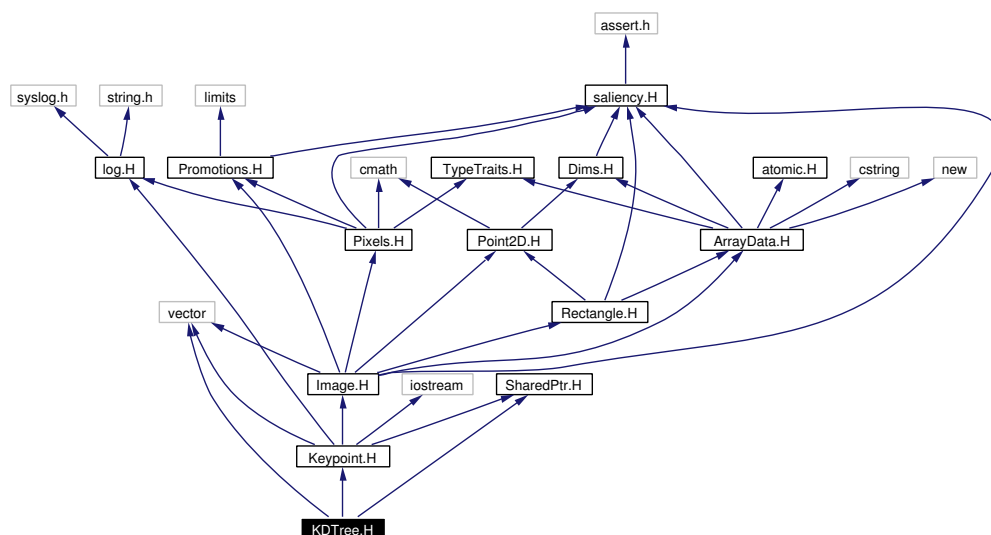
## 11.208 KDTree.H File Reference

```
#include "Keypoint.H"
```

```
#include "SharedPtr.H"
```

```
#include <vector>
```

Include dependency graph for KDTree.H:



This graph shows which files directly or indirectly include this file:

### Classes

- class [BestEntry](#)  
*The nearest entry in the tree to the target entry.*
- class [SortedLimitedList](#)  
*Maintains a sorted list with a limited number of elements.*
- class [HyperRectangle](#)  
*Represents a rectangle in n-dimensional space.*
- class [KDTree](#)  
*A simple K-d tree implementation.*



- class [HREntry](#)

*Represents a [HyperRectangle](#) entry in a list.*

### 11.208.1 Detailed Description

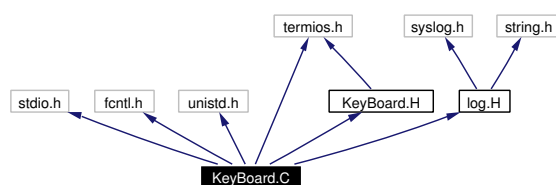
k-d tree implementation

Definition in file [KDTree.H](#).

## 11.209 KeyBoard.C File Reference

```
#include <stdio.h>
#include <fcntl.h>
#include <unistd.h>
#include <termios.h>
#include "KeyBoard.H"
#include "log.H"
```

Include dependency graph for KeyBoard.C:



### 11.209.1 Detailed Description

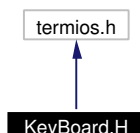
a simple keyboard interface

Definition in file [KeyBoard.C](#).

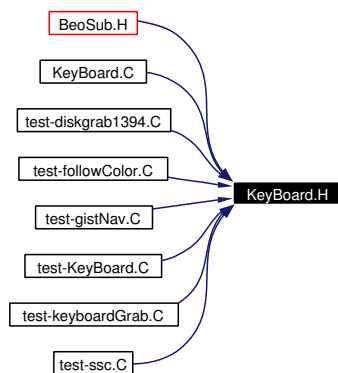
## 11.210 KeyBoard.H File Reference

```
#include <termios.h>
```

Include dependency graph for KeyBoard.H:



This graph shows which files directly or indirectly include this file:



### Classes

- class [KeyBoard](#)  
*Simple interface to a keyboard.*

### Enumerations

- enum [KeyBoardKey](#) {  
    **KBD\_NONE, KBD\_KEY1, KBD\_KEY2, KBD\_KEY3,**  
    **KBD\_KEY4, KBD\_KEY5, KBD\_OTHER }**  
    *Possible keys returned by our access methods.*

### 11.210.1 Detailed Description

a simple keyboard interface

Definition in file [KeyBoard.H](#).

### 11.210.2 Enumeration Type Documentation

#### 11.210.2.1 enum [KeyBoardKey](#)

Possible keys returned by our access methods.

Definition at line 43 of file KeyBoard.H.

Referenced by `KeyBoard::getKey()`, and `BeoSub::getKey()`.

## 11.211 Keypoint.C File Reference

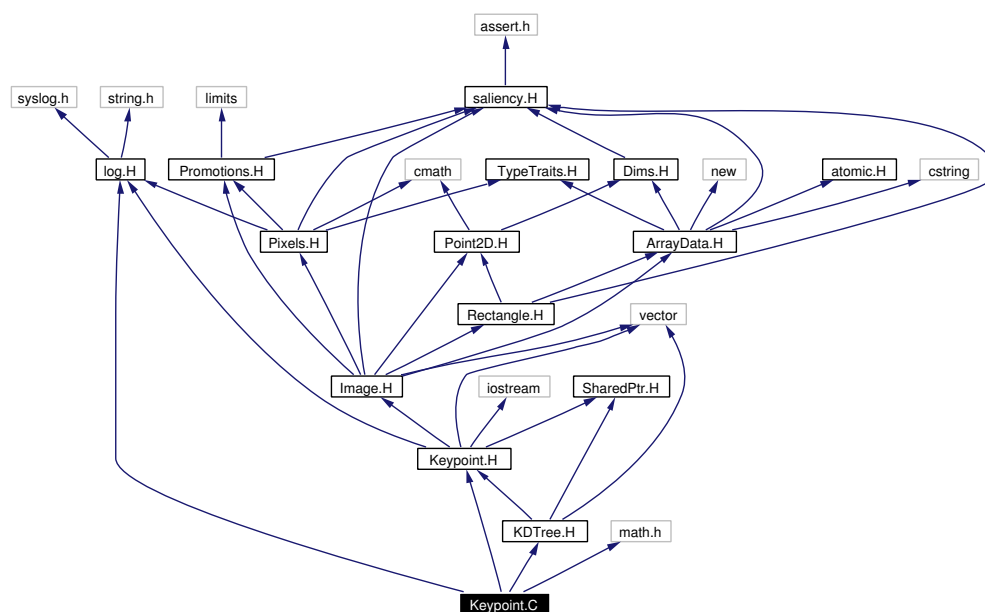
```
#include "Keypoint.H"
```

```
#include "KDTree.H"
```

```
#include "log.H"
```

```
#include <math.h>
```

Include dependency graph for Keypoint.C:



### 11.211.1 Detailed Description

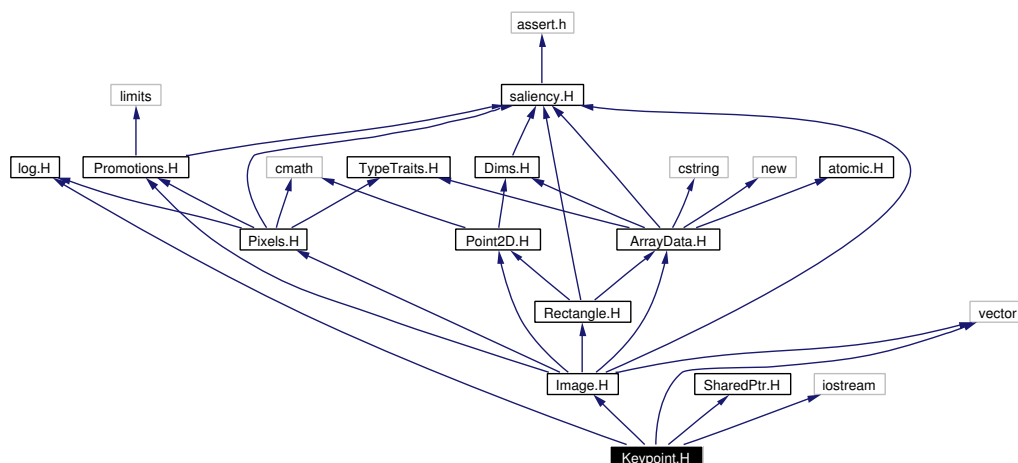
Keypoint for SIFT obj recognition

Definition in file [Keypoint.C](#).

## 11.212 Keypoint.H File Reference

```
#include "Image.H"
#include "SharedPtr.H"
#include "log.H"
#include <iostream>
#include <vector>
```

Include dependency graph for Keypoint.H:



This graph shows which files directly or indirectly include this file:

### Classes

- class **Keypoint**
- class [FeatureVector](#)

*Feature Vector.*

### 11.212.1 Detailed Description

Keypoint for SIFT obj recognition

Definition in file [Keypoint.H](#).

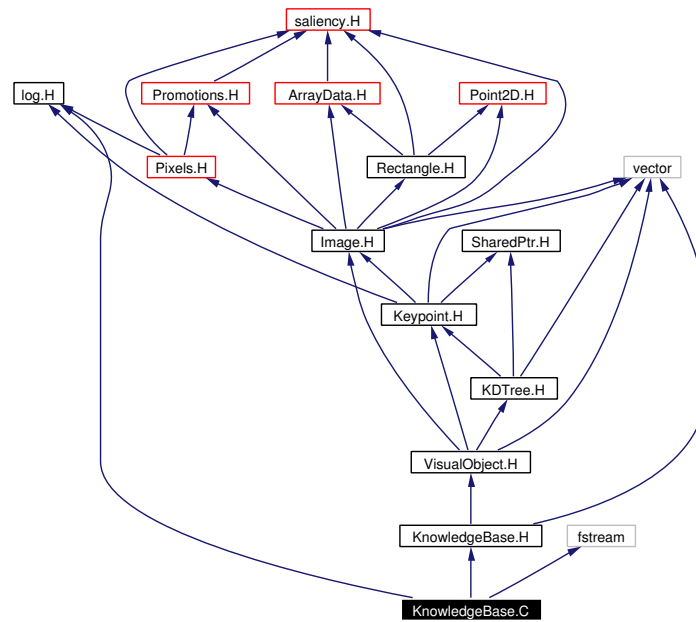
## 11.213 KnowledgeBase.C File Reference

```
#include "KnowledgeBase.H"
```

```
#include <fstream>
```

```
#include "log.H"
```

Include dependency graph for KnowledgeBase.C:



### 11.213.1 Detailed Description

Knowledge Base class

Definition in file [KnowledgeBase.C](#).

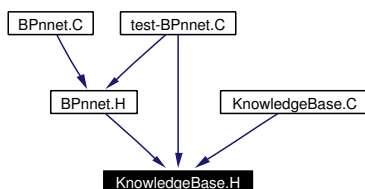
## 11.214 KnowledgeBase.H File Reference

```
#include "VisualObject.H"
```

```
#include <vector>
```

Include dependency graph for KnowledgeBase.H:

This graph shows which files directly or indirectly include this file:



### Classes

- class [KnowledgeBase](#)  
*this class holds a knowledge base*

### 11.214.1 Detailed Description

Header file for knowledge base

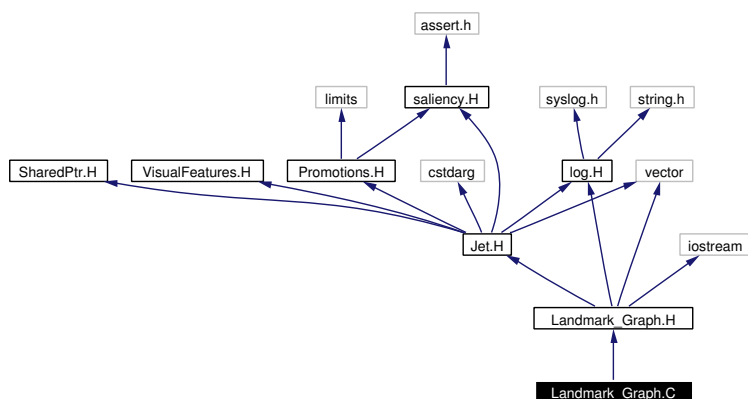
Definition in file [KnowledgeBase.H](#).



## 11.215 Landmark\_Graph.C File Reference

```
#include "Landmark_Graph.H"
```

Include dependency graph for Landmark\_Graph.C:



### 11.215.1 Detailed Description

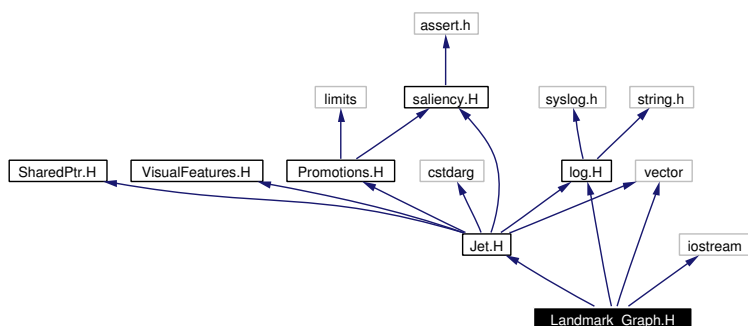
File to create and maintain landmark graph. Each graph is a vector of vertex, adjacency list pairs.

Definition in file [Landmark\\_Graph.C](#).

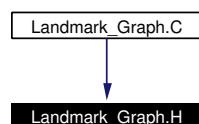
## 11.216 Landmark\_Graph.H File Reference

```
#include "Jet.H"
#include "log.H"
#include <vector>
#include <iostream>
```

Include dependency graph for Landmark\_Graph.H:



This graph shows which files directly or indirectly include this file:



## Classes

- class **Vertex**
- class [Edge](#)

*Definition of Edge class and its functions.*

- class **locn**
- class **LGraph**

### 11.216.1 Detailed Description

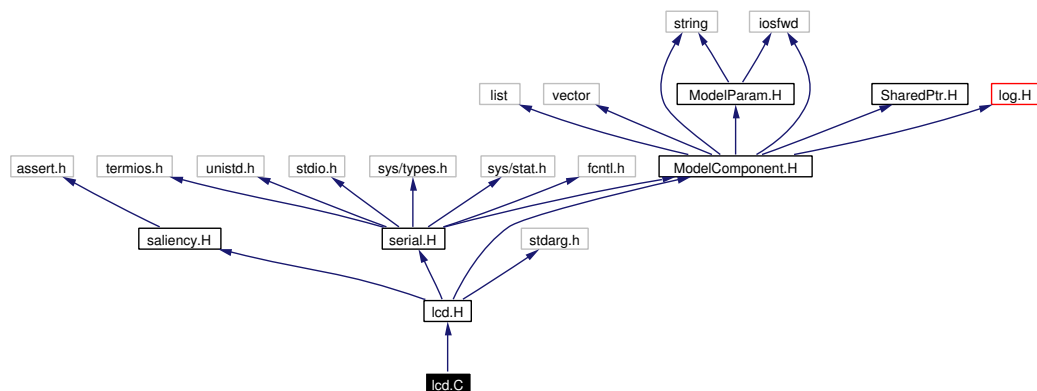
file to create and maintain landmark graph. Each graph is a vector of vertex, adjacency list pairs.

Definition in file [Landmark\\_Graph.H](#).

## 11.217 lcd.C File Reference

```
#include "lcd.H"
```

Include dependency graph for lcd.C:



### Defines

- `#define LCD_FUNCTION` 254  
*Char to be sent to gain access to special functions:.*

### 11.217.1 Detailed Description

Interface to an LCD screen via serial port

Definition in file [lcd.C](#).

### 11.217.2 Define Documentation

#### 11.217.2.1 `#define LCD_FUNCTION` 254

Char to be sent to gain access to special functions:.

Definition at line 53 of file [lcd.C](#).

```
#include "saliency.H"
#include "ModelComponent.H"
#include "serial.H"
#include <stdarg.h>
```

```

graph TD
    BeobotEffectors.C --> lcd.H[lcd.H]
    lcd.C --> lcd.H
    lcd.C --> lcd.H
    test-diskgrab1394.C --> lcd.H
    test-gistNav-slave-lcd.C --> lcd.H
    test-gistNav.C --> lcd.H
    test-ssc.C --> lcd.H
  
```

The diagram illustrates the inheritance of the `lcd.H` module. A central black box labeled `lcd.H` has seven arrows pointing to it from white boxes above. The white boxes are labeled: `BeobotEffectors.C`, `lcd.C` (appearing twice), `test-diskgrab1394.C`, `test-gistNav-slave-lcd.C`, `test-gistNav.C`, and `test-ssc.C`.

- class `lcd`  
*lcd.H* Interface to an LCD screen via serial port

Interface to an LCD screen via serial port  
Definition in file [lcd.H](#).

## 11.219 lcdecho.C File Reference

```
#include "lcd.H"  
#include "ModelManager.H"  
#include "ModelOptionDefs.H"  
#include <cstdlib>  
#include <iostream>
```

Include dependency graph for lcdecho.C:

### Functions

- int [main](#) (const int argc, const char \*argv[ ])

#### 11.219.1 Detailed Description

Clone of 'echo' but for an LCD screen on serial port

Definition in file [lcdecho.C](#).

#### 11.219.2 Function Documentation

##### 11.219.2.1 int main (const int *argc*, const char \* *argv*[ ])

This program provides basic interfacing to an LCD screen. It is intended for use in shell scripts. The basic usage is:

lcdecho p ... print the strings  
lcdecho c clear the screen  
lcdecho l move cursor left 1 or positions  
lcdecho r move cursor right 1 or positions  
lcdecho m set cursor position to

Definition at line 54 of file lcdecho.C.

References [main\(\)](#), [MYLOGVERB](#), and [uint](#).

## 11.220 LeakyIntegrator.H File Reference

This graph shows which files directly or indirectly include this file:



### Classes

- class [LeakyIntegrator](#)  
*A leaky integrator neuron, used by standard saliency map (SM).*

#### 11.220.1 Detailed Description

Class declarations for a leaky integrator neuron

Definition in file [LeakyIntegrator.H](#).

## 11.221 LeakyIntFire.H File Reference

This graph shows which files directly or indirectly include this file:



### Classes

- class [LeakyIntFire](#)  
*A leaky integrate & fire neuron, used by standard winner-take-all (WTA).*

#### 11.221.1 Detailed Description

Class declarations for a leaky integrator neuron

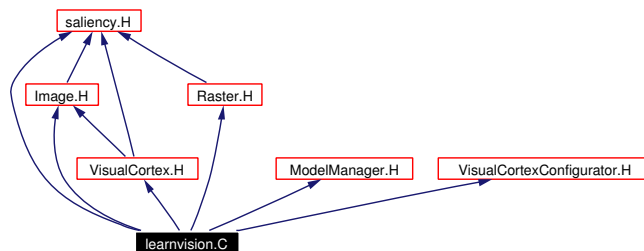
Definition in file [LeakyIntFire.H](#).



## 11.222 learnvision.C File Reference

```
#include "saliency.H"
#include "ModelManager.H"
#include "Image.H"
#include "VisualCortexConfigurator.H"
#include "VisualCortex.H"
#include "Raster.H"
```

Include dependency graph for learnvision.C:



## Functions

- `int main (const int argc, const char **argv)`  
Basic program to learn feature map weights from static images.

### 11.222.1 Detailed Description

like [ezvision.C](#) but focused on learning

Definition in file [learnvision.C](#).

### 11.222.2 Function Documentation

#### 11.222.2.1 `int main (const int argc, const char ** argv)`

Basic program to learn feature map weights from static images.

This program allows training of the relative weights of feature maps, given an image and associated binary target mask.

Definition at line 47 of file learnvision.C.

References `image`, `main()`, `MYLOGVERB`, `Raster::ReadGray()`, and `Raster::ReadRGB()`.

## 11.223 LevelSpec.C File Reference

```
#include "LevelSpec.H"
```

```
#include <stdio.h>
```

```
#include "saliency.H"
```

Include dependency graph for LevelSpec.C:

### 11.223.1 Detailed Description

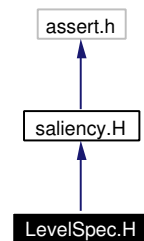
a utility class for use with [SingleChannel](#)

Definition in file [LevelSpec.C](#).

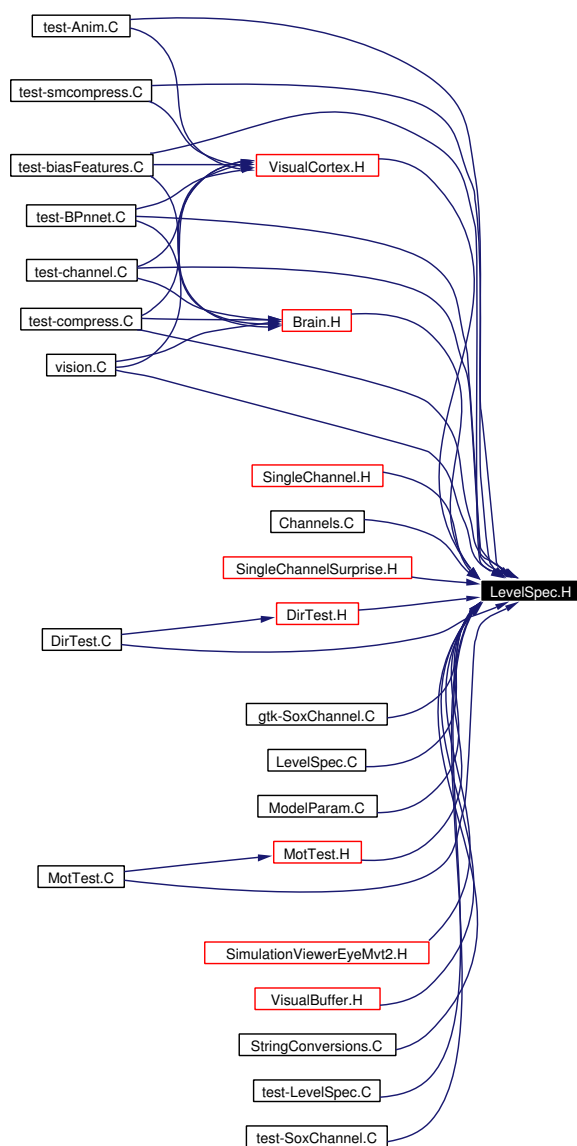
## 11.224 LevelSpec.H File Reference

```
#include "saliency.H"
```

Include dependency graph for LevelSpec.H:



This graph shows which files directly or indirectly include this file:



## Classes

- class [LevelSpec](#)

*A utility class for use with [SingleChannel](#).*

### 11.224.1 Detailed Description

a utility class for use with [SingleChannel](#)

Definition in file [LevelSpec.H](#).

## 11.225 log.C File Reference

```
#include "log.H"
#include <cerrno>
#include <cstdarg>
#include <cstdlib>
#include <stdio>
#include <sys/types.h>
#include <unistd.h>
#include <cstring>
```

Include dependency graph for log.C:

### Defines

- `#define LBS 1024`  
*total max size of a log entry*

### Functions

- `int snprintf (char *str, size_t n, const char *format,...)`
- `int vsnprintf (char *str, size_t n, const char *format, va_list ap)`
- `void sendlog (const char *msg,...)`  
*Send a log. NON-GNUC, thread-unsafe version.*

### Variables

- `bool FULLTRACE = true`
- `int MYLOGVERB = LOG_DEBUG`
- `char MYLOGFUN [LBS2]`
- `char MYLOGFIL [LBS2]`
- `int MYLOGLIN`
- `int MYLOGLEV`
- `int MYLOGIDVAL`
- `bool MYLOG_USEP`
- `bool MYLOG_USEID`

### 11.225.1 Detailed Description

Interface to syslog

Definition in file [log.C](#).

### 11.225.2 Define Documentation

#### 11.225.2.1 `#define LBS 1024`

total max size of a log entry

Definition at line 47 of file [log.C](#).

Referenced by [sendlog\(\)](#).

### 11.225.3 Function Documentation

#### 11.225.3.1 `void sendlog (const char * msg, ...)`

Send a log. NON-GNUC, thread-unsafe version.

Definition at line 153 of file [log.C](#).

References [FULLTRACE](#), [LBS](#), [LBS2](#), and [MYLOGVERB](#).

### 11.225.4 Variable Documentation

#### 11.225.4.1 `bool FULLTRACE = true`

Messages are sent to stderr if [FULLTRACE](#) is true, or are sent to syslog if [FULLTRACE](#) is false. Default value is true. Executables don't need to do anything special unless they need a non-default value.

Definition at line 49 of file [log.C](#).

Referenced by [main\(\)](#), and [sendlog\(\)](#).

#### 11.225.4.2 `int MYLOGVERB = LOG_DEBUG`

Log verbosity (see [<syslog.h>](#) for possible values). Default value is [LOG\\_DEBUG](#). Executables don't need to do anything special unless they need a non-default value.

Definition at line 50 of file [log.C](#).

Referenced by [main\(\)](#), [sendlog\(\)](#), and [ModelManager::setModelParamString\(\)](#).

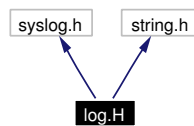


## 11.226 log.H File Reference

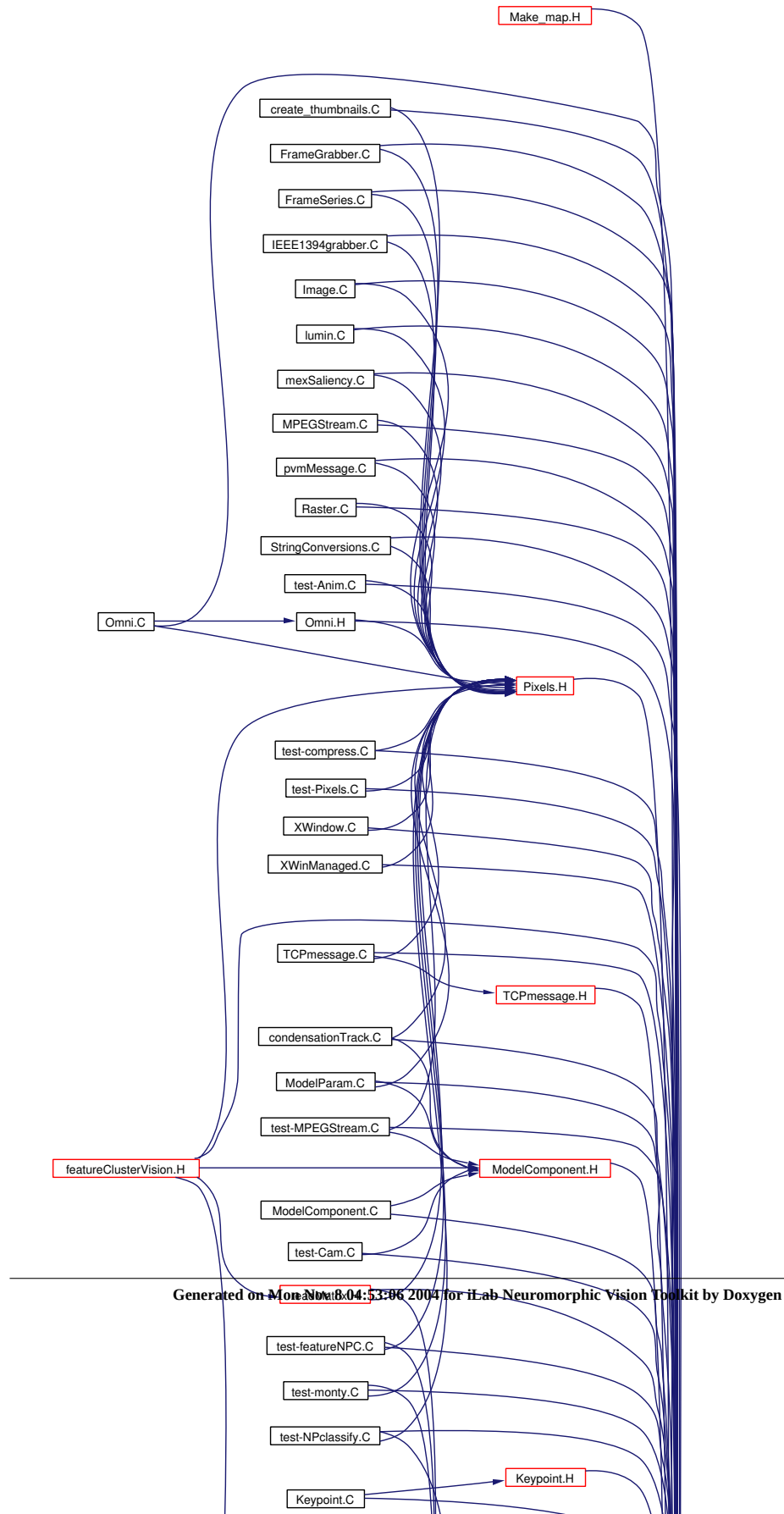
```
#include <syslog.h>
```

```
#include <string.h>
```

Include dependency graph for log.H:



This graph shows which files directly or indirectly include this file:



## Classes

- class [bug](#)

*This gets thrown by the FATAL logs.*

## Defines

- #define [LBS2](#) 64  
*max size of function or file name*
- #define **SENDLOG** MYLOG\_USEID=false, MYLOG\_USEP=false, sendlog
- #define **SENDPLOG** MYLOG\_USEID=false, MYLOG\_USEP=true, sendlog
- #define **SENDIDLOG** MYLOG\_USEID=true, MYLOG\_USEP=false, sendlog
- #define **SENDIDPLOG** MYLOG\_USEID=true, MYLOG\_USEP=true, sendlog
- #define **LDEBUG**
- #define **LINFO**
- #define **LERROR**
- #define **LFATAL**
- #define **PLDEBUG**
- #define **PLINFO**
- #define **PLError**
- #define **PLFATAL**
- #define **IDLDEBUG**
- #define **IDLINFO**
- #define **IDLError**
- #define **IDLFATAL**
- #define **IDPLDEBUG**
- #define **IDPLINFO**
- #define **IDPLError**
- #define **IDPLFATAL**
- #define **CLOGFILE** ("[" + descriptiveName() + "]).c\_str()
- #define **CLDEBUG**
- #define **CLINFO**
- #define **CLERROR**
- #define **CLFATAL**
- #define **CPLDEBUG**
- #define **CPLINFO**
- #define **CPLERROR**
- #define **CPLFATAL**

## Functions

- void [sendlog](#) (const char \*msg,...)  
*Send a log. NON-GNUC, thread-unsafe version.*

## Variables

- bool [FULLTRACE](#)
- int [MYLOGVERB](#)
- char [MYLOGFUN](#) []
- char [MYLOGFIL](#) []
- int [MYLOGLIN](#)
- int [MYLOGLEV](#)
- int [MYLOGIDVAL](#)
- bool [MYLOG\\_USEID](#)
- bool [MYLOG\\_USEP](#)

### 11.226.1 Detailed Description

Error message facility and interface to syslog

Definition in file [log.H](#).

### 11.226.2 Define Documentation

#### 11.226.2.1 #define CLDEBUG

**Value:**

```
MYLOGLIN=__LINE__,strncpy(MYLOGFUN, __FUNCTION__, LBS2-1),\  
strncpy(MYLOGFIL, CLOGFILE, LBS2-1),MYLOGLEV=LOG_DEBUG,SENDLOG
```

Definition at line 271 of file log.H.

#### 11.226.2.2 #define CLERROR

**Value:**

```
MYLOGLIN=__LINE__,strncpy(MYLOGFUN, __FUNCTION__, LBS2-1),\  
strncpy(MYLOGFIL, CLOGFILE, LBS2-1),MYLOGLEV=LOG_ERR,SENDLOG
```

Definition at line 275 of file log.H.

**11.226.2.3 #define CLFATAL****Value:**

```
MYLOGLIN=__LINE__,strncpy(MYLOGFUN, __FUNCTION__, LBS2-1),\  
strncpy(MYLOGFIL, CLOGFILE, LBS2-1),MYLOGLEV=LOG_CRIT,SENDLOG
```

Definition at line 277 of file log.H.

**11.226.2.4 #define CLINFO****Value:**

```
MYLOGLIN=__LINE__,strncpy(MYLOGFUN, __FUNCTION__, LBS2-1),\  
strncpy(MYLOGFIL, CLOGFILE, LBS2-1),MYLOGLEV=LOG_INFO,SENDLOG
```

Definition at line 273 of file log.H.

**11.226.2.5 #define CPLDEBUG****Value:**

```
MYLOGLIN=__LINE__,strncpy(MYLOGFUN, __FUNCTION__, LBS2-1),\  
strncpy(MYLOGFIL, CLOGFILE, LBS2-1),MYLOGLEV=LOG_DEBUG,SENDPLOG
```

Definition at line 280 of file log.H.

**11.226.2.6 #define CPLERROR****Value:**

```
MYLOGLIN=__LINE__,strncpy(MYLOGFUN, __FUNCTION__, LBS2-1),\  
strncpy(MYLOGFIL, CLOGFILE, LBS2-1),MYLOGLEV=LOG_ERR,SENDPLOG
```

Definition at line 284 of file log.H.

**11.226.2.7 #define CPLFATAL****Value:**

```
MYLOGLIN=__LINE__,strncpy(MYLOGFUN, __FUNCTION__, LBS2-1),\  
strncpy(MYLOGFIL, CLOGFILE, LBS2-1),MYLOGLEV=LOG_CRIT,SENDPLOG
```

Definition at line 286 of file log.H.

**11.226.2.8 #define CPLINFO****Value:**

```
MYLOGLIN=__LINE__,strncpy(MYLOGFUN, __FUNCTION__, LBS2-1),\  
strncpy(MYLOGFIL, CLOGFILE, LBS2-1),MYLOGLEV=LOG_INFO,SENDPLOG
```

Definition at line 282 of file log.H.

**11.226.2.9 #define IDLDEBUG****Value:**

```
MYLOGLIN=__LINE__,strncpy(MYLOGFUN, __FUNCTION__, LBS2-1),\  
MYLOGIDVAL=MYLOGID,strncpy(MYLOGFIL, __FILE__, LBS2-1),MYLOGLEV=LOG_DEBUG,\  
SENDIDLOG
```

Definition at line 243 of file log.H.

**11.226.2.10 #define IDLError****Value:**

```
MYLOGLIN=__LINE__,strncpy(MYLOGFUN, __FUNCTION__, LBS2-1),\  
MYLOGIDVAL=MYLOGID,strncpy(MYLOGFIL, __FILE__, LBS2-1),MYLOGLEV=LOG_ERR,\  
SENDIDLOG
```

Definition at line 249 of file log.H.

**11.226.2.11 #define IDLFATAL****Value:**

```
MYLOGLIN=__LINE__,strncpy(MYLOGFUN, __FUNCTION__, LBS2-1),\  
MYLOGIDVAL=MYLOGID,strncpy(MYLOGFIL, __FILE__, LBS2-1),MYLOGLEV=LOG_CRIT,\  
SENDIDLOG
```

Definition at line 252 of file log.H.

**11.226.2.12 #define IDLINFO****Value:**

```
MYLOGLIN=__LINE__,strncpy(MYLOGFUN, __FUNCTION__, LBS2-1),\
MYLOGIDVAL=MYLOGID,strncpy(MYLOGFIL, __FILE__, LBS2-1),MYLOGLEV=LOG_INFO,\
SENDIDLOG
```

Definition at line 246 of file log.H.

#### 11.226.2.13 #define IDPLDEBUG

**Value:**

```
MYLOGLIN=__LINE__,strncpy(MYLOGFUN, __FUNCTION__, LBS2-1),\
MYLOGIDVAL=MYLOGID,strncpy(MYLOGFIL, __FILE__, LBS2-1),MYLOGLEV=LOG_DEBUG,\
SENDIDPLOG
```

Definition at line 256 of file log.H.

#### 11.226.2.14 #define IDPLERROR

**Value:**

```
MYLOGLIN=__LINE__,strncpy(MYLOGFUN, __FUNCTION__, LBS2-1),\
MYLOGIDVAL=MYLOGID,strncpy(MYLOGFIL, __FILE__, LBS2-1),MYLOGLEV=LOG_ERR,\
SENDIDPLOG
```

Definition at line 262 of file log.H.

#### 11.226.2.15 #define IDPLFATAL

**Value:**

```
MYLOGLIN=__LINE__,strncpy(MYLOGFUN, __FUNCTION__, LBS2-1),\
MYLOGIDVAL=MYLOGID,strncpy(MYLOGFIL, __FILE__, LBS2-1),MYLOGLEV=LOG_CRIT,\
SENDIDPLOG
```

Definition at line 265 of file log.H.

#### 11.226.2.16 #define IDPLINFO

**Value:**

```
MYLOGLIN=__LINE__,strncpy(MYLOGFUN, __FUNCTION__, LBS2-1),\
MYLOGIDVAL=MYLOGID,strncpy(MYLOGFIL, __FILE__, LBS2-1),MYLOGLEV=LOG_INFO,\
SENDIDPLOG
```

Definition at line 259 of file log.H.

**11.226.2.17 #define LBS2 64**

max size of function or file name

Definition at line 217 of file log.H.

Referenced by sendlog().

**11.226.2.18 #define LDEBUG****Value:**

```
MYLOGLIN=__LINE__,strncpy(MYLOGFUN, __FUNCTION__, LBS2-1),\
strncpy(MYLOGFIL, __FILE__, LBS2-1),MYLOGLEV=LOG_DEBUG,SENDLOG
```

Definition at line 225 of file log.H.

**11.226.2.19 #define LERROR****Value:**

```
MYLOGLIN=__LINE__,strncpy(MYLOGFUN, __FUNCTION__, LBS2-1),\
strncpy(MYLOGFIL, __FILE__, LBS2-1),MYLOGLEV=LOG_ERR,SENDLOG
```

Definition at line 229 of file log.H.

**11.226.2.20 #define LFATAL****Value:**

```
MYLOGLIN=__LINE__,strncpy(MYLOGFUN, __FUNCTION__, LBS2-1),\
strncpy(MYLOGFIL, __FILE__, LBS2-1),MYLOGLEV=LOG_CRIT,SENDLOG
```

Definition at line 231 of file log.H.

**11.226.2.21 #define LINFO****Value:**

```
MYLOGLIN=__LINE__,strncpy(MYLOGFUN, __FUNCTION__, LBS2-1),\
strncpy(MYLOGFIL, __FILE__, LBS2-1),MYLOGLEV=LOG_INFO,SENDLOG
```

Definition at line 227 of file log.H.



**11.226.2.22 #define PLDEBUG****Value:**

```
MYLOGLIN=__LINE__,strncpy(MYLOGFUN, __FUNCTION__, LBS2-1),\  
strncpy(MYLOGFIL, __FILE__, LBS2-1),MYLOGLEV=LOG_DEBUG,SENDPLOG
```

Definition at line 234 of file log.H.

**11.226.2.23 #define PLERROR****Value:**

```
MYLOGLIN=__LINE__,strncpy(MYLOGFUN, __FUNCTION__, LBS2-1),\  
strncpy(MYLOGFIL, __FILE__, LBS2-1),MYLOGLEV=LOG_ERR,SENDPLOG
```

Definition at line 238 of file log.H.

**11.226.2.24 #define PLFATAL****Value:**

```
MYLOGLIN=__LINE__,strncpy(MYLOGFUN, __FUNCTION__, LBS2-1),\  
strncpy(MYLOGFIL, __FILE__, LBS2-1),MYLOGLEV=LOG_CRIT,SENDPLOG
```

Definition at line 240 of file log.H.

**11.226.2.25 #define PLINFO****Value:**

```
MYLOGLIN=__LINE__,strncpy(MYLOGFUN, __FUNCTION__, LBS2-1),\  
strncpy(MYLOGFIL, __FILE__, LBS2-1),MYLOGLEV=LOG_INFO,SENDPLOG
```

Definition at line 236 of file log.H.

**11.226.3 Function Documentation****11.226.3.1 void sendlog (const char \* *msg*, ...)**

Send a log. NON-GNUC, thread-unsafe version.

Definition at line 153 of file log.C.

References FULLTRACE, LBS, LBS2, and MYLOGVERB.

## 11.226.4 Variable Documentation

### 11.226.4.1 bool **FULLTRACE**

Messages are sent to stderr if FULLTRACE is true, or are sent to syslog if FULLTRACE is false. Default value is true. Executables don't need to do anything special unless they need a non-default value.

Definition at line 132 of file log.H.

Referenced by main(), and sendlog().

### 11.226.4.2 int **MYLOGVERB**

Log verbosity (see <syslog.h> for possible values). Default value is LOG\_DEBUG. Executables don't need to do anything special unless they need a non-default value.

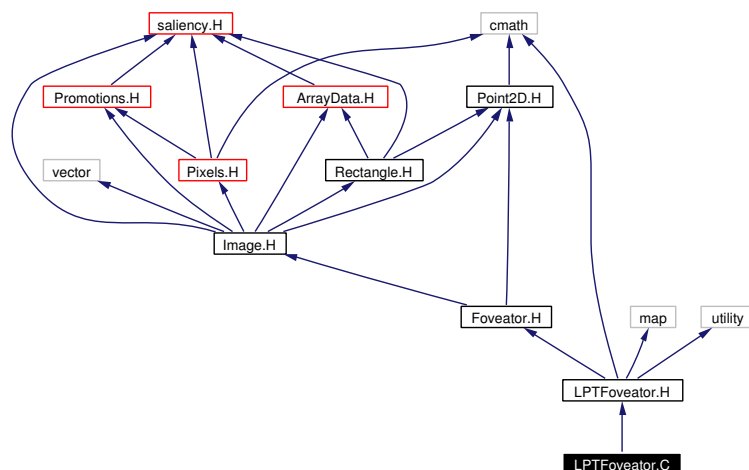
Definition at line 137 of file log.H.

Referenced by main(), sendlog(), and ModelManager::setModelParamString().

## 11.227 LPTFoveator.C File Reference

```
#include "LPTFoveator.H"
```

Include dependency graph for LPTFoveator.C:



### 11.227.1 Detailed Description

[Foveator](#) class that performs log polar transform

Definition in file [LPTFoveator.C](#).

## 11.228 LPTFoveator.H File Reference

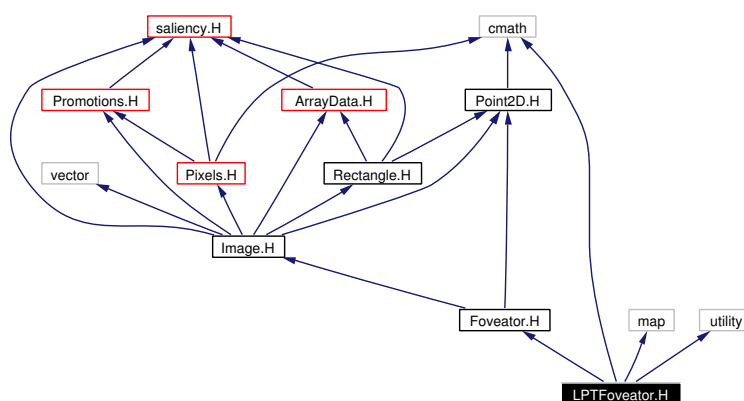
```
#include "Foveator.H"
```

```
#include <map>
```

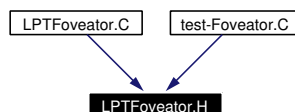
```
#include <utility>
```

```
#include <cmath>
```

Include dependency graph for LPTFoveator.H:



This graph shows which files directly or indirectly include this file:



## Classes

- class [LPTFoveator](#)

*A class that does space-variant processing with log polar transform.*

- struct **LPTFoveator::ltPoint2D**

*a function object for the key compare in multimap*

### 11.228.1 Detailed Description

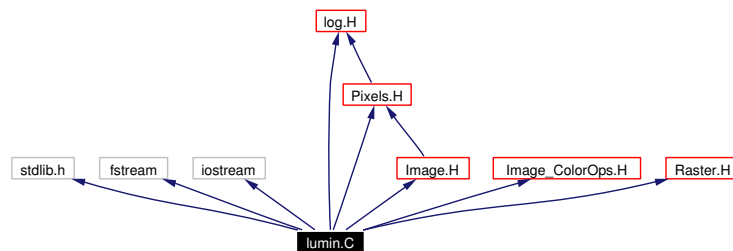
[Foveator](#) class that performs log polar transform

Definition in file [LPTFoveator.H](#).

## 11.229 lumin.C File Reference

```
#include <stdlib.h>
#include <fstream>
#include <iostream>
#include "log.H"
#include "Image.H"
#include "Image_ColorOps.H"
#include "Raster.H"
#include "Pixels.H"
```

Include dependency graph for lumin.C:



## Functions

- int **main** (int argc, char \*argv[ ])

## Variables

- char \* **filename**  
*this is a simple program for determining the average luminosity of an imagewhat file to open*
- **Image**< **PixRGB**< float > > **image**  
*image object*
- **Image**< **PixRGB**< byte > > **bimage**  
*byte based image*
- **Image**< float > **fimage**

*float image B/W*

- float [lum](#)

*luminance of image*

### 11.229.1 Detailed Description

NERD CAM program sweet

Definition in file [lumin.C](#).

### 11.229.2 Variable Documentation

#### 11.229.2.1 [Image](#)<[PixRGB](#) <[byte](#)> > [bimage](#)

byte based image

Definition at line 66 of file lumin.C.

#### 11.229.2.2 [char\\*](#) [filename](#)

this is a simple program for determining the average luminosity of an imagewhat file to open

Definition at line 62 of file lumin.C.

#### 11.229.2.3 [Image](#)<[float](#)> [fimage](#)

float image B/W

Definition at line 68 of file lumin.C.

#### 11.229.2.4 [Image](#)<[PixRGB](#) <[float](#)> > [image](#)

image object

Definition at line 64 of file lumin.C.

Referenced by `TemplateMatchPyrBuilder::build()`, `ReichardtPyrBuilder< T >::build()`, `GenericPyrBuilder< T >::build()`, `PsychoDisplay::displayImagePatch()`, `XWinManaged::drawImage()`, `SimulationViewerStd::drawTime()`, `contourRun::dump-EnergySigmoid()`, `V4Lgrabber::grab()`, `IEEE1394grabber::grab()`, `V4Lgrabber::grab-Single()`, `ScaleSpace::initScaleSpace()`, `main()`, `PsychoDisplay::makeBlittable-Surface()`, `Hmax::origGetC2()`, `Omni< T >::run()`, `Raster::Visu()`, `Raster::Visu-`

Array(), Raster::WriteFloat(), OutputFrameSeries::writeFloat(), Raster::WriteFloatNorm(), Raster::WriteGray(), OutputFrameSeries::writeGray(), Raster::WriteRGB(), and OutputFrameSeries::writeRGB().

#### 11.229.2.5 float **lum**

luminance of image

Definition at line 70 of file lumin.C.

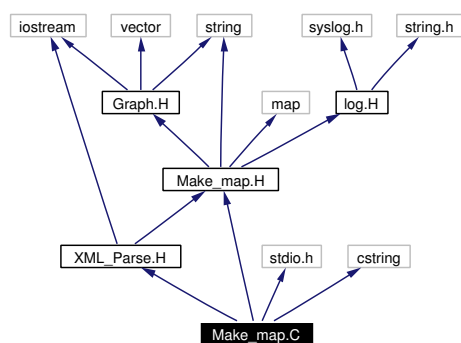
Referenced by getRGBY(), Image< T >::luminanceNormalize(), BeobotVisualCortex::masterProcess(), and PixRGB< T >::PixRGB().



## 11.230 Make\_map.C File Reference

```
#include "Make_map.H"  
#include "XML_Parse.H"  
#include <stdio.h>  
#include <cstring>
```

Include dependency graph for Make\_map.C:



### 11.230.1 Detailed Description

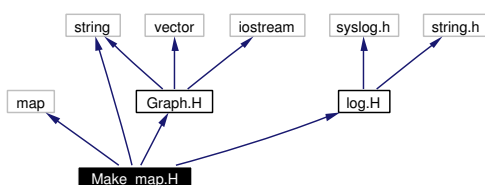
This has methods of [Map](#) class

Definition in file [Make\\_map.C](#).

## 11.231 Make\_map.H File Reference

```
#include <map>
#include <string>
#include "log.H"
#include "Graph.H"
```

Include dependency graph for Make\_map.H:



This graph shows which files directly or indirectly include this file:

### Classes

- class [Map](#)

*Definition of Map class and access functions.*

### Typedefs

- typedef [std::map](#)< [std::string](#), [Entity](#), std::less< [std::string](#) > > **map\_ontology**
- typedef [std::map](#)< [std::string](#), [Entity\\_TWM](#), std::less< [std::string](#) > > **map\_twm**

### 11.231.1 Detailed Description

converted to a multi level hash table as ontology grows in future

Definition in file [Make\\_map.H](#).

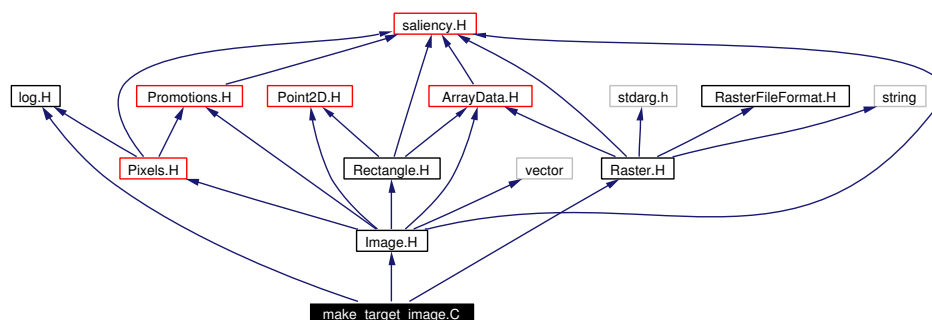
## 11.232 make\_target\_image.C File Reference

```
#include "Image.H"
```

```
#include "log.H"
```

```
#include "Raster.H"
```

Include dependency graph for make\_target\_image.C:



### Functions

- `int main (int argc, char **argv)`

#### 11.232.1 Detailed Description

reads PNM w/255=target; removes all other colors

Definition in file [make\\_target\\_image.C](#).

## 11.233 make\_test\_movie.C File Reference

```
#include "saliency.H"
#include "log.H"
#include "Image_All.H"
#include "Raster.H"
```

Include dependency graph for make\_test\_movie.C:

### Functions

- int [main](#) (int argc, char \*\*argv)

### 11.233.1 Detailed Description

make a simple test movie

Definition in file [make\\_test\\_movie.C](#).

### 11.233.2 Function Documentation

#### 11.233.2.1 int main (int *argc*, char \*\* *argv*)

This program creates a simple movie in which a blob smoothly drifts down/right, then abruptly jumps to the left side, and finally drifts up/right. This can be used to test saccade controllers. Input image should be 640x480 for best results.

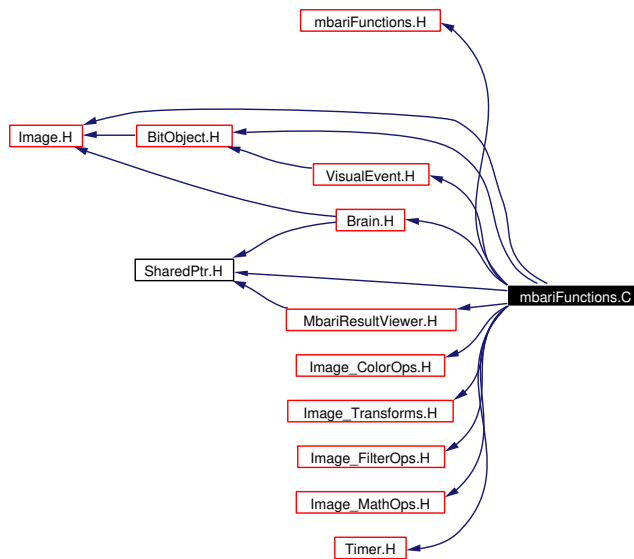
Definition at line 47 of file make\_test\_movie.C.

References [Image< T >::drawDisk\(\)](#), [Point2D::i](#), [Point2D::j](#), [main\(\)](#), [Raster::ReadRGB\(\)](#), and [Raster::WriteRGB\(\)](#).

## 11.234 mbariFunctions.C File Reference

```
#include "mbariFunctions.H"
#include "VisualEvent.H"
#include "BitObject.H"
#include "Image.H"
#include "MbariResultViewer.H"
#include "SharedPtr.H"
#include "Image_ColorOps.H"
#include "Image_Transforms.H"
#include "Image_FilterOps.H"
#include "Image_MathOps.H"
#include "Brain.H"
#include "Timer.H"
```

Include dependency graph for mbariFunctions.C:



## Functions

- `std::list< BitObject > extractBitObjects (const Image< byte > &blmg, Rectangle region, int minSize)`

*extract a set of BitObjects from bitImg, which intersect region*

- `std::list< BitObject > getLargestObjects` (const `Image< byte >` &bitImg, `Rectangle` region, int numObjects, int minSize)
- `std::list< BitObject > getSalRegions` (`SharedPtr< Brain >` brain, const `Image< PixRGB< byte > >` &img, const `Image< byte >` &bitImg, float maxEvolveTime, int maxNumSalSpots, int minSize)

*returns the BitObjects at the most salient points*

- `Image< byte > showAllObjects` (`std::list< BitObject >` &objs)

*all BitObjects in objs are drawn into the return image*

### 11.234.1 Detailed Description

a few functions used by MBARI programs

Definition in file [mbariFunctions.C](#).

### 11.234.2 Function Documentation

#### 11.234.2.1 `std::list<BitObject> extractBitObjects` (const `Image< byte >` & `bitImg`, `Rectangle` region, int `minSize` = 0)

extract a set of BitObjects from bitImg, which intersect region

bitImg is flooded starting from each point within region, and each `BitObject` exceeding the minSize is stored in the list of BitObjects that is returned.

Definition at line 53 of file [mbariFunctions.C](#).

References `Rectangle::bottom()`, `byte`, `Rectangle::fitToDims()`, `Timer::get()`, `BitObject::getArea()`, `Image< T >::getDims()`, `Image< T >::getVal()`, `Rectangle::left()`, `replaceVals()`, `Timer::reset()`, `BitObject::reset()`, `Rectangle::right()`, `takeMax()`, `Rectangle::top()`, and `ZEROS`.

Referenced by `getSalRegions()`, and `VisualEventSet::updateEvents()`.

#### 11.234.2.2 `std::list<BitObject> getSalRegions` (`SharedPtr< Brain >` brain, const `Image< PixRGB< byte > >` &img, const `Image< byte >` & `bitImg`, float `maxEvolveTime`, int `maxNumSalSpots`, int `minSize` = 0)

returns the BitObjects at the most salient points

#### Parameters:

**brain** the brain object used for the saliency computation

*img* the image scanned for salient locations

*bitImg* a binary image used for extracting BitObjects at the salient locations

*maxEvolveTime* the maximum evolution time for the saliency map

*maxNumSalSpots* the maximum number of salient locations extracted

*the* minimum size of a [BitObject](#) to be considered

Definition at line 133 of file mbariFunctions.C.

References [extractBitObjects\(\)](#), [Image< T >::getDims\(\)](#), [Point2D::i](#), [Point2D::j](#), and [SharedPtr< T >::reset\(\)](#).

#### 11.234.2.3 [Image<byte>](#) [showAllObjects](#) ([std::list< BitObject >](#) & *objs*)

all BitObjects in *objs* are drawn into the return image

Definition at line 209 of file mbariFunctions.C.

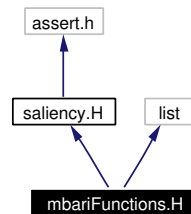
References [byte](#), [Image< T >::initialized\(\)](#), and [takeMax\(\)](#).

## 11.235 mbariFunctions.H File Reference

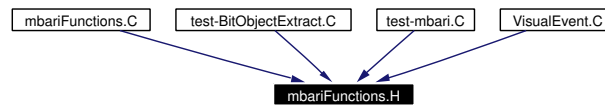
```
#include "saliency.H"
```

```
#include <list>
```

Include dependency graph for mbariFunctions.H:



This graph shows which files directly or indirectly include this file:



## Functions

- `std::list< BitObject > extractBitObjects` (const `Image< byte > &bitImg`, `Rectangle` region, int minSize=0)

*extract a set of BitObjects from bitImg, which intersect region*

- `std::list< BitObject > getLargestObjects` (const `Image< byte > &bitImg`, `Rectangle` region, int numObjects, int minSize)
- `std::list< BitObject > getSalRegions` (`SharedPtr< Brain > brain`, const `Image< PixRGB< byte > > &img`, const `Image< byte > &bitImg`, float maxEvolveTime, int maxNumSalSpots, int minSize=0)

*returns the BitObjects at the most salient points*

- `Image< byte > showAllObjects` (`std::list< BitObject > &objs`)

*all BitObjects in objs are drawn into the return image*



### 11.235.1 Detailed Description

a few functions used by MBARI programs

Definition in file [mbariFunctions.H](#).

### 11.235.2 Function Documentation

#### 11.235.2.1 `std::list<BitObject> extractBitObjects (const Image< byte > & bitImg, Rectangle region, int minSize = 0)`

extract a set of BitObjects from bitImg, which intersect region

bitImg is flooded starting from each point within region, and each BitObject exceeding the minSize is stored in the list of BitObjects that is returned.

Definition at line 53 of file mbariFunctions.C.

References [Rectangle::bottom\(\)](#), [byte](#), [Rectangle::fitToDims\(\)](#), [Timer::get\(\)](#), [BitObject::getArea\(\)](#), [Image< T >::getDims\(\)](#), [Image< T >::getVal\(\)](#), [Rectangle::left\(\)](#), [replaceVals\(\)](#), [BitObject::reset\(\)](#), [Timer::reset\(\)](#), [Rectangle::right\(\)](#), [takeMax\(\)](#), [Rectangle::top\(\)](#), and [ZEROS](#).

Referenced by [getSalRegions\(\)](#), and [VisualEventSet::updateEvents\(\)](#).

#### 11.235.2.2 `std::list<BitObject> getSalRegions (SharedPtr< Brain > brain, const Image< PixRGB< byte > > & img, const Image< byte > & bitImg, float maxEvolveTime, int maxNumSalSpots, int minSize = 0)`

returns the BitObjects at the most salient points

#### Parameters:

**brain** the brain object used for the saliency computation

**img** the image scanned for salient locations

**bitImg** a binary image used for extracting BitObjects at the salient locations

**maxEvolveTime** the maximum evolution time for the saliency map

**maxNumSalSpots** the maximum number of salient locations extracted

**the** minimum size of a BitObject to be considered

Definition at line 133 of file mbariFunctions.C.

References [extractBitObjects\(\)](#), [Image< T >::getDims\(\)](#), [Point2D::i](#), [Point2D::j](#), and [SharedPtr< T >::reset\(\)](#).

**11.235.2.3   `Image<byte>` `showAllObjects` (`std::list< BitObject >` & *objs*)**

all BitObjects in *objs* are drawn into the return image

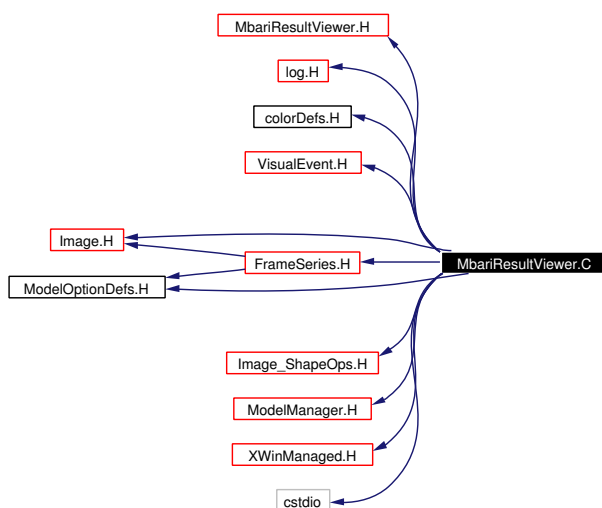
Definition at line 209 of file `mbariFunctions.C`.

References `byte`, `Image< T >::initialized()`, and `takeMax()`.

## 11.236 MbariResultViewer.C File Reference

```
#include "MbariResultViewer.H"  
#include "log.H"  
#include "colorDefs.H"  
#include "VisualEvent.H"  
#include "Image.H"  
#include "Image_ShapeOps.H"  
#include "ModelManager.H"  
#include "XWinManaged.H"  
#include "FrameSeries.H"  
#include "ModelOptionDefs.H"  
#include <stdio>
```

Include dependency graph for MbariResultViewer.C:



## Defines

- #define INSTITUTE(T)

## Functions

- `INSTANTIATE` (`PixRGB`< `byte` >)
- `INSTANTIATE` (`byte`)
- `INSTANTIATE` (`float`)

### 11.236.1 Detailed Description

class that manages the results viewing and saving for the MBARI programs

Definition in file [MbariResultViewer.C](#).

### 11.236.2 Define Documentation

#### 11.236.2.1 `#define INSTANTIATE(T)`

**Value:**

```
template void MbariResultViewer::output(const Image< T >& img, \
                                         const uint frameNum, \
                                         const std::string& resultName, \
                                         const int resNum); \
template void MbariResultViewer::display(const Image< T >& img, \
                                         const uint frameNum, \
                                         const std::string& resultName, \
                                         const int resNum); \
template XWinManaged* MbariResultViewer::displayImage(const Image< T >& img, \
                                                         XWinManaged* win, \
                                                         const char* label);
```

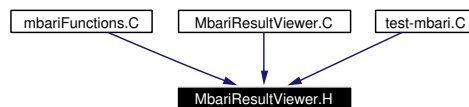
Definition at line 534 of file [MbariResultViewer.C](#).

## 11.237 MbariResultViewer.H File Reference

```
#include "saliency.H"
#include "ModelComponent.H"
#include "SharedPtr.H"
#include "Pixels.H"
#include "ArrayData.H"
#include "BitObjectDrawModes.H"
#include <string>
```

Include dependency graph for MbariResultViewer.H:

This graph shows which files directly or indirectly include this file:



### Classes

- class [MbariResultViewer](#)  
*Class to facilitate displaying and saving of intermediate results.*

#### 11.237.1 Detailed Description

class that manages the results viewing and saving for the MBARI programs

Definition in file [MbariResultViewer.H](#).

## 11.238 mexConverts.C File Reference

```
#include "mexLog.H"
#include "mexConverts.H"
#include "SharedPtr.H"
#include "Image.H"
```

Include dependency graph for mexConverts.C:

### Defines

- #define **INstantiate**(T)
- #define **INstantiate2**(T)

### Functions

- template<class T> [Image](#)< T > [mexArray2Image](#) (const mxArray \*arrptr)  
*convert a mxArray into a grayscale image*
- template<class T> [Image](#)< [PixRGB](#)< T > > [mexArray2RGBImage](#) (const mxArray \*arrptr)  
*convert a mxArray into a RGB image*
- template<class T> mxArray \* [Image2mexArray](#) (const [Image](#)< T > &img)  
*convert a grayscale image into a mxArray*
- template<class T> mxArray \* [RGBImage2mexArray](#) (const [Image](#)< [PixRGB](#)< T > > &img)  
*convert a RGB image into a mxArray*
- template<class T> [std::vector](#)< T > [mexArr2Vector](#) (const mxArray \*vecptr, [uint](#) num\_elements)  
*convert the first num\_elements elements of a mxArray into a [std::vector](#)*
- template<class T> [std::vector](#)< T > [mexArr2Vector](#) (const mxArray \*vecptr)  
*convert a mxArray vector into a [std::vector](#)*
- template<class T> mxArray \* [Vector2mexArr](#) (const [std::vector](#)< T > &vect, [uint](#) num\_elements)  
*convert the first num\_elements elements of a [std::vector](#) into a mxArray*

- `template<class T> mxArray * Vector2mexArr (const std::vector< T > &vect)`  
convert a [std::vector](#) into a mxArray vector
- `std::vector< Point2D > mexArr2Point2DVec (const mxArray *arrptr)`  
convert a 2 x N mxArray into a [std::vector](#)<[Point2D](#)>
- `mxArray * Point2DVec2mexArr (const std::vector< Point2D > &vect)`  
convert a [std::vector](#)<[Point2D](#)> into a 2 x N mxArray
- `template<class T> mxArray * ImgVec2mexArr (const ImageSet< T > &vect)`  
convert a vector of pointers to images of equal size into a 3D mxArray
- `template<class T> ImageSet< T > mexArr2ImgVec (const mxArray *arrptr)`  
convert a 3D mxArray into a vector of pointers to images of equal size
- `template<class T> mxArray * ImageSet2cellArr (const ImageSet< T > &imset)`  
convert an [ImageSet](#) into a cell array of images
- `template<class T> ImageSet< T > cellArr2ImageSet (const mxArray *arrptr)`  
convert a cell array of images into an [ImageSet](#)
- `template<class T> mxArray * ImageSetVec2cellArr (const std::vector< ImageSet< T > > &vec)`  
convert a vector of [ImageSet](#) into a 2d cell array
- `template<class T> std::vector< ImageSet< T > > cellArr2ImageSetVec (const mxArray *arrptr)`  
convert a 2d cell array into a vector of [ImageSet](#)
- **INstantiate** ([byte](#))
- **INstantiate** ([float](#))
- **INstantiate2** ([byte](#))
- **INstantiate2** ([float](#))
- **INstantiate2** ([int](#))
- **INstantiate2** ([double](#))

### 11.238.1 Detailed Description

conversions from and to MEX arrays

Definition in file [mexConverts.C](#).

## 11.238.2 Define Documentation

### 11.238.2.1 #define INSTANTIATE(T)

**Value:**

```
template Image<T> mxArray2Image(const mxArray* arrptr); \
template Image< PixRGB<T> > mxArray2RGBImage(const mxArray* arrptr); \
template mxArray* Image2mexArray(const Image<T>& img); \
template mxArray* RGBImage2mexArray(const Image< PixRGB<T> >& img); \
template mxArray* ImgVec2mexArr(const ImageSet<T>& vect); \
template ImageSet<T> mexArr2ImgVec(const mxArray* arrptr); \
template mxArray* ImageSet2cellArr(const ImageSet<T>& imset); \
template ImageSet<T> cellArr2ImageSet(const mxArray* arrptr); \
template mxArray* ImageSetVec2cellArr(const std::vector< ImageSet<T> >& vec); \
template std::vector< ImageSet<T> > cellArr2ImageSetVec(const mxArray* arrptr); \
template bool MexReturn::store(int num, const Image<T>& val); \
template bool MexReturn::store(int num, const Image< PixRGB<T> >& val); \
```

Definition at line 568 of file mexConverts.C.

### 11.238.2.2 #define INSTANTIATE2(T)

**Value:**

```
template std::vector<T> mexArr2Vector(const mxArray* vecptr, uint num_elements); \
template std::vector<T> mexArr2Vector(const mxArray* vecptr); \
template mxArray* Vector2mexArr(const std::vector<T>& vect, uint num_elements); \
template mxArray* Vector2mexArr(const std::vector<T>& vect); \
template bool MexReturn::store(int num, const std::vector<T>& val); \
```

Definition at line 586 of file mexConverts.C.

## 11.238.3 Function Documentation

### 11.238.3.1 template<class T> [ImageSet](#)<T> cellArr2ImageSet (const mxArray \* arrptr)

convert a cell array of images into an [ImageSet](#)

Definition at line 437 of file mexConverts.C.

References [mexFatal\(\)](#).

### 11.238.3.2 template<class T> [std::vector](#)< [ImageSet](#)<T> > cellArr2ImageSetVec (const mxArray \* arrptr)

convert a 2d cell array into a vector of [ImageSet](#)



Definition at line 471 of file mexConverts.C.

References mexFatal(), and uint.

### 11.238.3.3 `template<class T> mxArray* Image2mexArray (const Image< T > & img)`

convert a grayscale image into a mxArray

Definition at line 168 of file mexConverts.C.

References Image< T >::begin(), Image< T >::getHeight(), Image< T >::getSize(), and Image< T >::getWidth().

Referenced by ImageSet2cellArr(), ImageSetVec2cellArr(), and MexReturn::store().

### 11.238.3.4 `template<class T> mxArray* ImageSet2cellArr (const ImageSet< T > & imset)`

convert an ImageSet into a cell array of images

Definition at line 424 of file mexConverts.C.

References Image2mexArray(), and ImageSet< T >::size().

### 11.238.3.5 `template<class T> mxArray* ImageSetVec2cellArr (const std::vector< ImageSet< T > > & vec)`

convert a vector of ImageSet into a 2d cell array

Definition at line 452 of file mexConverts.C.

References Image2mexArray().

### 11.238.3.6 `template<class T> mxArray* ImgVec2mexArr (const ImageSet< T > & vect)`

convert a vector of pointers to images of equal size into a 3D mxArray

Definition at line 305 of file mexConverts.C.

References ImageSet< T >::isEmpty(), mexFatal(), and ImageSet< T >::size().

### 11.238.3.7 `template<class T> ImageSet<T> mexArr2ImgVec (const mxArray * arrptr)`

convert a 3D mxArray into a vector of pointers to images of equal size

Definition at line 341 of file mexConverts.C.

References mexFatal(), and NO\_INIT.

#### 11.238.3.8 `std::vector<Point2D> mexArr2Point2DVec (const mxArray * arrptr)`

convert a 2 x N mxArray into a `std::vector<Point2D>`

Definition at line 268 of file mexConverts.C.

References mexFatal().

#### 11.238.3.9 `template<class T> std::vector<T> mexArr2Vector (const mxArray * vecptr)`

convert a mxArray vector into a `std::vector`

Definition at line 238 of file mexConverts.C.

References uint.

#### 11.238.3.10 `template<class T> std::vector<T> mexArr2Vector (const mxArray * vecptr, uint num_elements)`

convert the first num\_elements elements of a mxArray into a `std::vector`

Definition at line 221 of file mexConverts.C.

#### 11.238.3.11 `template<class T> Image<T> mxArray2Image (const mxArray * arrptr)`

convert a mxArray into a grayscale image

Definition at line 45 of file mexConverts.C.

References Image< T >::beginw(), mexFatal(), and NO\_INIT.

#### 11.238.3.12 `template<class T> Image< PixRGB<T> > mxArray2RGBImage (const mxArray * arrptr)`

convert a mxArray into a RGB image

Definition at line 100 of file mexConverts.C.

References Image< T >::beginw(), mexFatal(), NO\_INIT, PixRGB< T >::setBlue(), PixRGB< T >::setGreen(), and PixRGB< T >::setRed().

**11.238.3.13** `mxArray* Point2DVec2mexArr (const std::vector< Point2D > & vect)`

convert a [std::vector](#)<[Point2D](#)> into a 2 x N mxArray

Definition at line 290 of file mexConverts.C.

References [uint](#).

**11.238.3.14** `template<class T> mxArray* RGBImage2mexArray (const Image< PixRGB< T > > & img)`

convert a RGB image into a mxArray

Definition at line 191 of file mexConverts.C.

References [PixRGB](#)< T >::blue(), [PixRGB](#)< T >::green(), and [PixRGB](#)< T >::red().

Referenced by [MexReturn::store\(\)](#).

**11.238.3.15** `template<class T> mxArray* Vector2mexArr (const std::vector< T > & vect)`

convert a [std::vector](#) into a mxArray vector

Definition at line 262 of file mexConverts.C.

Referenced by [MexReturn::store\(\)](#).

**11.238.3.16** `template<class T> mxArray* Vector2mexArr (const std::vector< T > & vect, uint num_elements)`

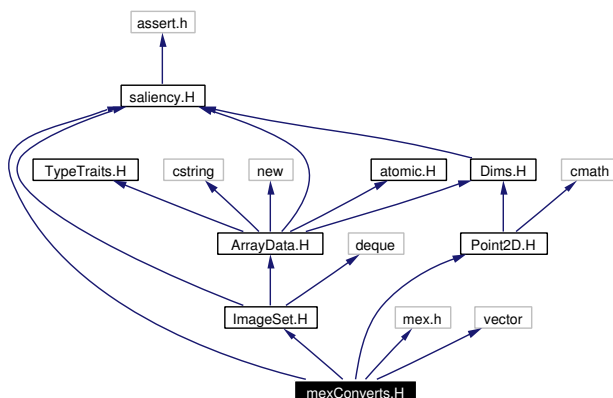
convert the first num\_elements elements of a [std::vector](#) into a mxArray

Definition at line 245 of file mexConverts.C.

## 11.239 mexConverts.H File Reference

```
#include "saliency.H"
#include "Point2D.H"
#include "ImageSet.H"
#include <mex.h>
#include <vector>
```

Include dependency graph for mexConverts.H:



This graph shows which files directly or indirectly include this file:

### Classes

- class [MexReturn](#)  
*Class to help manage return values in mex files.*

### Functions

- template<class T> [Image](#)< T > [mexArray2Image](#) (const mxArray \*arrptr)  
*convert a mxArray into a grayscale image*
- template<class T> [Image](#)< [PixRGB](#)< T > > [mexArray2RGBImage](#) (const mxArray \*arrptr)  
*convert a mxArray into a RGB image*

- `template<class T> mxArray * Image2mexArray (const Image< T > &img)`  
*convert a grayscale image into a mxArray*
- `template<class T> mxArray * RGBImage2mexArray (const Image< PixRGB< T > > &img)`  
*convert a RGB image into a mxArray*
- `template<class T> std::vector< T > mexArr2Vector (const mxArray *vecptr, uint num_elements)`  
*convert the first num\_elements elements of a mxArray into a std::vector*
- `template<class T> std::vector< T > mexArr2Vector (const mxArray *vecptr)`  
*convert a mxArray vector into a std::vector*
- `template<class T> mxArray * Vector2mexArr (const std::vector< T > &vect, uint num_elements)`  
*convert the first num\_elements elements of a std::vector into a mxArray*
- `template<class T> mxArray * Vector2mexArr (const std::vector< T > &vect)`  
*convert a std::vector into a mxArray vector*
- `std::vector< Point2D > mexArr2Point2DVec (const mxArray *arrptr)`  
*convert a 2 x N mxArray into a std::vector<Point2D>*
- `mxArray * Point2DVec2mexArr (const std::vector< Point2D > &vect)`  
*convert a std::vector<Point2D> into a 2 x N mxArray*
- `template<class T> mxArray * ImgVec2mexArr (const ImageSet< T > &vect)`  
*convert a vector of pointers to images of equal size into a 3D mxArray*
- `template<class T> ImageSet< T > mexArr2ImgVec (const mxArray *arrptr)`  
*convert a 3D mxArray into a vector of pointers to images of equal size*
- `template<class T> mxArray * ImageSet2cellArr (const ImageSet< T > &im-set)`  
*convert an ImageSet into a cell array of images*
- `template<class T> ImageSet< T > cellArr2ImageSet (const mxArray *arrptr)`  
*convert a cell array of images into an ImageSet*
- `template<class T> mxArray * ImageSetVec2cellArr (const std::vector< Image-Set< T > > &vec)`

*convert a vector of [ImageSet](#) into a 2d cell array*

- `template<class T> std::vector< ImageSet< T > > cellArr2ImageSetVec (const mxArray *arrptr)`

*convert a 2d cell array into a vector of [ImageSet](#)*

- `double conv (const double *src, double *target)`
- `byte conv (const double *src, byte *target)`
- `double conv (const byte *src, double *target)`
- `byte conv (const byte *src, byte *target)`
- `float conv (const double *src, float *target)`
- `double conv (const float *src, double *target)`
- `float conv (const byte *src, float *target)`

### 11.239.1 Detailed Description

conversions from ans to MEX arrays

Definition in file [mexConverts.H](#).

### 11.239.2 Function Documentation

#### 11.239.2.1 `template<class T> ImageSet<T> cellArr2ImageSet (const mxArray * arrptr)`

convert a cell array of images into an [ImageSet](#)

Definition at line 437 of file [mexConverts.C](#).

References [mexFatal\(\)](#).

#### 11.239.2.2 `template<class T> std::vector< ImageSet<T> > cellArr2ImageSetVec (const mxArray * arrptr)`

convert a 2d cell array into a vector of [ImageSet](#)

Definition at line 471 of file [mexConverts.C](#).

References [mexFatal\(\)](#), and [uint](#).

#### 11.239.2.3 `template<class T> mxArray* Image2mexArray (const Image< T > & img)`

convert a grayscale image into a `mxArray`

Definition at line 168 of file mexConverts.C.

References `Image< T >::begin()`, `Image< T >::getHeight()`, `Image< T >::getSize()`, and `Image< T >::getWidth()`.

Referenced by `ImageSet2cellArr()`, `ImageSetVec2cellArr()`, and `MexReturn::store()`.

#### 11.239.2.4 `template<class T> mxArray* ImageSet2cellArr (const ImageSet< T > & imset)`

convert an `ImageSet` into a cell array of images

Definition at line 424 of file mexConverts.C.

References `Image2mexArray()`, and `ImageSet< T >::size()`.

#### 11.239.2.5 `template<class T> mxArray* ImageSetVec2cellArr (const std::vector< ImageSet< T > > & vec)`

convert a vector of `ImageSet` into a 2d cell array

Definition at line 452 of file mexConverts.C.

References `Image2mexArray()`.

#### 11.239.2.6 `template<class T> mxArray* ImgVec2mexArr (const ImageSet< T > & vect)`

convert a vector of pointers to images of equal size into a 3D `mexArray`

Definition at line 305 of file mexConverts.C.

References `ImageSet< T >::isEmpty()`, `mexFatal()`, and `ImageSet< T >::size()`.

#### 11.239.2.7 `template<class T> ImageSet<T> mexArr2ImgVec (const mxArray * arrptr)`

convert a 3D `mexArray` into a vector of pointers to images of equal size

Definition at line 341 of file mexConverts.C.

References `mexFatal()`, and `NO_INIT`.

#### 11.239.2.8 `std::vector<Point2D> mexArr2Point2DVec (const mxArray * arrptr)`

convert a 2 x N `mexArray` into a `std::vector<Point2D>`

Definition at line 268 of file mexConverts.C.

References mexFatal().

**11.239.2.9** `template<class T> std::vector<T> mexArr2Vector (const mxArray * vecptr)`

convert a mxArray vector into a [std::vector](#)

Definition at line 238 of file mexConverts.C.

References uint.

**11.239.2.10** `template<class T> std::vector<T> mexArr2Vector (const mxArray * vecptr, uint num_elements)`

convert the first num\_elements elements of a mxArray into a [std::vector](#)

Definition at line 221 of file mexConverts.C.

References uint.

**11.239.2.11** `template<class T> Image<T> mxArray2Image (const mxArray * arrptr)`

convert a mxArray into a grayscale image

Definition at line 45 of file mexConverts.C.

References [Image< T >::beginw\(\)](#), mexFatal(), and NO\_INIT.

**11.239.2.12** `template<class T> Image< PixRGB<T> > mxArray2RGBImage (const mxArray * arrptr)`

convert a mxArray into a RGB image

Definition at line 100 of file mexConverts.C.

References [Image< T >::beginw\(\)](#), mexFatal(), NO\_INIT, [PixRGB< T >::setBlue\(\)](#), [PixRGB< T >::setGreen\(\)](#), and [PixRGB< T >::setRed\(\)](#).

**11.239.2.13** `mxArray* Point2DVec2mexArr (const std::vector< Point2D > & vect)`

convert a [std::vector<Point2D>](#) into a 2 x N mxArray

Definition at line 290 of file mexConverts.C.



References uint.

**11.239.2.14** `template<class T> mxArray* RGBImage2mexArray (const Image< PixRGB< T > > & img)`

convert a RGB image into a mxArray

Definition at line 191 of file mexConverts.C.

References PixRGB< T >::blue(), PixRGB< T >::green(), and PixRGB< T >::red().

Referenced by MexReturn::store().

**11.239.2.15** `template<class T> mxArray* Vector2mexArr (const std::vector< T > & vect)`

convert a `std::vector` into a mxArray vector

Definition at line 262 of file mexConverts.C.

Referenced by MexReturn::store().

**11.239.2.16** `template<class T> mxArray* Vector2mexArr (const std::vector< T > & vect, uint num_elements)`

convert the first num\_elements elements of a `std::vector` into a mxArray

Definition at line 245 of file mexConverts.C.

References mexFatal(), and uint.

## 11.240 mexDummy.C File Reference

### 11.240.1 Detailed Description

- This does nothing at all. It is just there to appease the mex linker

Definition in file [mexDummy.C](#).

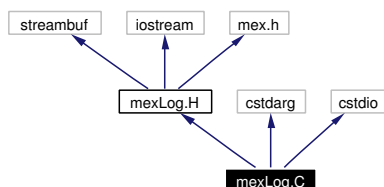
## 11.241 mexLog.C File Reference

```
#include "mexLog.H"
```

```
#include <cstdarg>
```

```
#include <cstdio>
```

Include dependency graph for mexLog.C:



### Defines

- #define **FPEXCEPTIONSDISABLED**
- #define **BUFMAXSIZE** 1000

### Functions

- void [mexInfo](#) (const char \*fmt,...)  
*writes an info message to mexPrintf*
- void [mexError](#) (const char \*fmt,...)  
*writes an error message*
- void [mexFatal](#) (const char \*fmt,...)  
*writes an error message and leaves the mex file*
- void [mexDebug](#) (const char \*fmt,...)  
*writes a debug message*

#### 11.241.1 Detailed Description

a few helpful functions for logging in mex files

Definition in file [mexLog.C](#).

## 11.241.2 Function Documentation

### 11.241.2.1 void mexDebug (const char \* *fmt*, ...)

writes a debug message

Definition at line 79 of file mexLog.C.

### 11.241.2.2 void mexError (const char \* *fmt*, ...)

writes an error message

Definition at line 57 of file mexLog.C.

Referenced by MexModelManager::parseCommandLine(), and MexReturn::~MexReturn().

### 11.241.2.3 void mexFatal (const char \* *fmt*, ...)

writes an error message and leaved the mex file

Definition at line 68 of file mexLog.C.

Referenced by cellArr2ImageSet(), cellArr2ImageSetVec(), MexModelManager::exitProgram(), MexModelManager::getExtraMexArg(), ImgVec2mexArr(), mexArr2ImgVec(), mexArr2Point2DVec(), mexArray2Image(), mexArray2RGBImage(), MexModelManager::printUsage(), Vector2mexArr(), and MexReturn::~MexReturn().

### 11.241.2.4 void mexInfo (const char \* *fmt*, ...)

writes an info message to mexPrintf

Definition at line 46 of file mexLog.C.

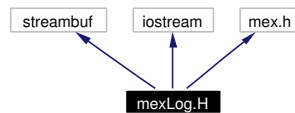
## 11.242 mexLog.H File Reference

```
#include <streambuf>
```

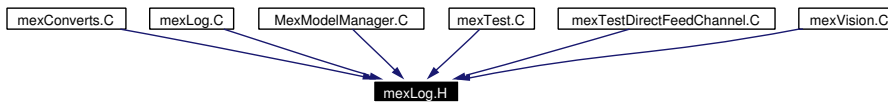
```
#include <iostream>
```

```
#include <mex.h>
```

Include dependency graph for mexLog.H:



This graph shows which files directly or indirectly include this file:



### Classes

- class [MexBuf](#)  
*This streambuf is used to re-direct cout and cerr.*
- class [MexBufInit](#)  
*This class triggers the redirection of the standard streams at construction.*

### Functions

- void [mexInfo](#) (const char \*fmt,...)  
*writes an info message to mexPrintf*
- void [mexError](#) (const char \*fmt,...)  
*writes an error message*
- void [mexFatal](#) (const char \*fmt,...)  
*writes an error message and leaved the mex file*

- void [mexDebug](#) (const char \*fmt,...)  
*writes a debug message*

## Variables

- [MexBuf](#) `mexbuf__`
- [MexBufInit](#) `mexbufinit__` (`mexbuf__`)

### 11.242.1 Detailed Description

a few helpful functions for logging in mex files

Definition in file [mexLog.H](#).

### 11.242.2 Function Documentation

#### 11.242.2.1 void mexDebug (const char \* *fmt*, ...)

writes a debug message

Definition at line 79 of file `mexLog.C`.

#### 11.242.2.2 void mexError (const char \* *fmt*, ...)

writes an error message

Definition at line 57 of file `mexLog.C`.

Referenced by `MexModelManager::parseCommandLine()`, and `MexReturn::~MexReturn()`.

#### 11.242.2.3 void mexFatal (const char \* *fmt*, ...)

writes an error message and leaved the mex file

Definition at line 68 of file `mexLog.C`.

Referenced by `cellArr2ImageSet()`, `cellArr2ImageSetVec()`, `MexModelManager::exitProgram()`, `MexModelManager::getExtraMexArg()`, `ImgVec2mexArr()`, `mexArr2ImgVec()`, `mexArr2Point2DVec()`, `mexArray2Image()`, `mexArray2RGBImage()`, `MexModelManager::printUsage()`, `Vector2mexArr()`, and `MexReturn::~MexReturn()`.

**11.242.2.4 void mexInfo (const char \* *fmt*, ...)**

writes an info message to mexPrintf

Definition at line 46 of file mexLog.C.

## 11.243 MexModelManager.C File Reference

```
#include "mexLog.H"  
#include "MexModelManager.H"  
#include "ModelOptionDefs.H"  
#include "fpe.H"
```

Include dependency graph for MexModelManager.C:

### 11.243.1 Detailed Description

Definition in file [MexModelManager.C](#).



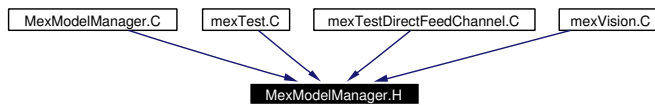
## 11.244 MexModelManager.H File Reference

```
#include "ModelManager.H"
```

```
#include <mex.h>
```

Include dependency graph for MexModelManager.H:

This graph shows which files directly or indirectly include this file:



### Classes

- class [MexModelManager](#)  
*A special form of [ModelManager](#) used in mex files.*

### 11.244.1 Detailed Description

Definition in file [MexModelManager.H](#).

## 11.245 mexSaliency.C File Reference

```
#include "mexConverts.H"
#include "saliency.H"
#include "log.H"
#include "fancynorm.H"
#include "getSaliency.H"
#include "Image.H"
#include "Point2D.H"
#include "Image_All.H"
#include "Channels.H"
#include "SharedPtr.H"
#include "Pixels.H"
#include <vector>
#include <string>
#include <cmath>
#include <sstream>
#include <mex.h>
#include "StdBrain.H"
#include "ModelManager.H"
#include "ModelOptionDefs.H"
#include "StringConversions.H"
#include "ShapeEstimator.H"
```

Include dependency graph for mexSaliency.C:

### Functions

- void **mexFunction** (int nlhs, mxArray \*plhs[ ], int nrhs, const mxArray \*prhs[ ])

#### 11.245.1 Detailed Description

This is the MEX file version of the Saliency program

This file compiles into a MEX file that can be called from within Matlab for extracting the saliency information from an image. The call from Matlab is:

```
[num_sal, coords, times, salmap, modfunc, areas, labels]
= Saliency(image, targets, max_time, foa_size, weights,
normtype, smoothFactor, levels);
```

The return values are:

**Parameters:**

- num\_sal** the number of salient spots found
- coords** (2, num\_sal) array that contains the x and y coordinates of the salient spots in (1,:) and (2,:), respectively
- times** (1, num\_sal) array that contains the evolution times
- salmap** the saliency map (normalized between 0 and 1)
- modfunc** (num\_sal, height, width) array that contains the modulation functions for the salient spots, maxnormalized to 1.0
- areas** (1, num\_sal) array that contains the number of pixels for salient spot that have contributed to the modulation function
- labels** {num\_sal} cell array containing the label strings from the size info analysis

The only *required* argument is:

**Parameters:**

- image** the input image

*Optional* paramters that have pre-assigned default values are:

**Parameters:**

- targets** map of the same size as the input image, in which targets for the focus of attention are 255 and the rest is zero. Pass a scalar 0 (the default) if you have no targets to look for.
- max\_time** the maximum amount of (simulated) time that the saliency map should evolve in seconds (*default: 0.7*)
- foa\_size** the size of the focus of attention in pixels. The default is 1/12 of min(height, width) of the input image. Pass -1 if you want to use the default.
- weights** Vector of length 3 containing the weights for the following channels (in this order): [wIntens, wOrient, wColor] (*default: [1.0 1.0 1.0]*)
- normtype** normalization type; see [fancynorm.H](#) (*default: 2 = VCXNORM\_NORM*)
- smoothMethod** method used for smoothing the shapeEstimator masks; see [ShapeEstimatorModes.H](#); (*default: 1 = Gaussian Smoothing*)

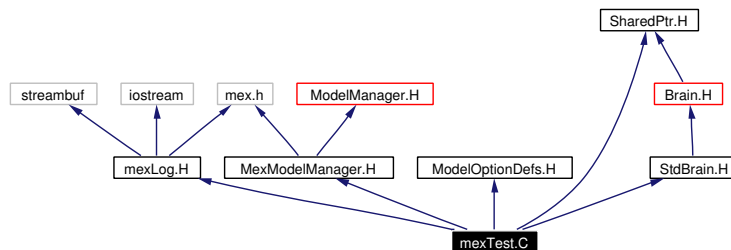
**levels** Vector of length 6 containing the following parameters: [sm\_level, level\_min, level\_max, delta\_min, delta\_max, nborients] (*default: [4 2 4 3 4 4]*)

Definition in file [mexSaliency.C](#).

## 11.246 mexTest.C File Reference

```
#include "mexLog.H"
#include "MexModelManager.H"
#include "ModelOptionDefs.H"
#include "SharedPtr.H"
#include "StdBrain.H"
```

Include dependency graph for mexTest.C:



### Functions

- void **mexFunction** (int nlhs, mxArray \*plhs[ ], int nrhs, const mxArray \*prhs[ ])

#### 11.246.1 Detailed Description

Test some mex functions

```
////////////////////////////////////// // The iLab Neuromorphic Vision C++
Toolkit - Copyright (C) 2001 by the // University of Southern California (USC)
and the iLab at USC. // See http://iLab.usc.edu for information about this
project. // //////////////////////////////////////// // Major portions of the
iLab Neuromorphic Vision Toolkit are protected // under the U.S. patent "Com-
putation of Intrinsic Perceptual Saliency // in Visual Environments, and Applica-
tions" by Christof Koch and // Laurent Itti, California Institute of Technology,
2001 (patent // pending; application number 09/912,225 filed July 23, 2001; see //
http://pair.uspto.gov/cgi-bin/final/home.pl for current status). //
////////////////////////////////////// // This file is part of the iLab Neuro-
morphic Vision C++ Toolkit. // // The iLab Neuromorphic Vision C++ Toolkit is free
software; you can // redistribute it and/or modify it under the terms of the GNU General
// Public License as published by the Free Software Foundation; either // version 2 of
```

the License, or (at your option) any later version. // // The iLab Neuromorphic Vision C++ Toolkit is distributed in the hope // that it will be useful, but WITHOUT ANY WARRANTY; without even the // implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR // PURPOSE. See the GNU General Public License for more details. // // You should have received a copy of the GNU General Public License // along with the iLab Neuromorphic Vision C++ Toolkit; if not, write // to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, // Boston, MA 02111-1307 USA. // ////////////////////////////////////// //

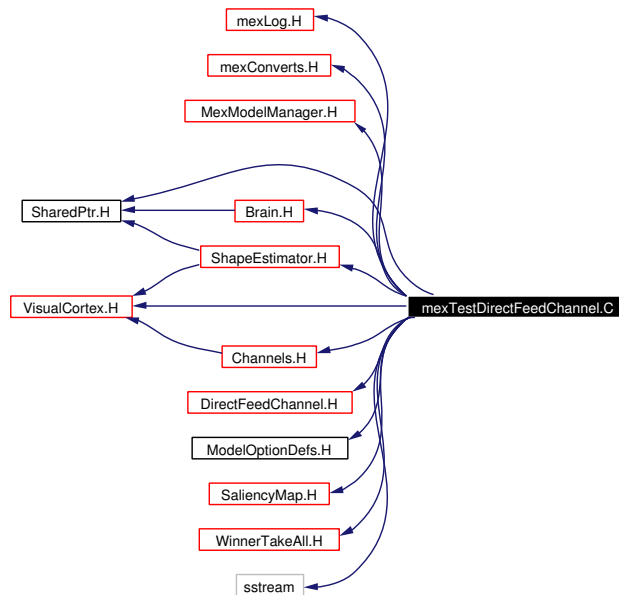
Primary maintainer for this file: Dirk Walther <[walther@caltech.edu](mailto:walther@caltech.edu)>

Definition in file [mexTest.C](#).

## 11.247 mexTestDirectFeedChannel.C File Reference

```
#include "mexLog.H"  
#include "mexConverts.H"  
#include "MexModelManager.H"  
#include "SharedPtr.H"  
#include "DirectFeedChannel.H"  
#include "ModelOptionDefs.H"  
#include "VisualCortex.H"  
#include "Brain.H"  
#include "SaliencyMap.H"  
#include "WinnerTakeAll.H"  
#include "Channels.H"  
#include "ShapeEstimator.H"  
#include <sstream>
```

Include dependency graph for mexTestDirectFeedChannel.C:



## Functions

- void **mexFunction** (int nlhs, mxArray \*plhs[ ], int nrhs, const mxArray \*prhs[ ])

### 11.247.1 Detailed Description

- test the [DirectFeedChannel](#)

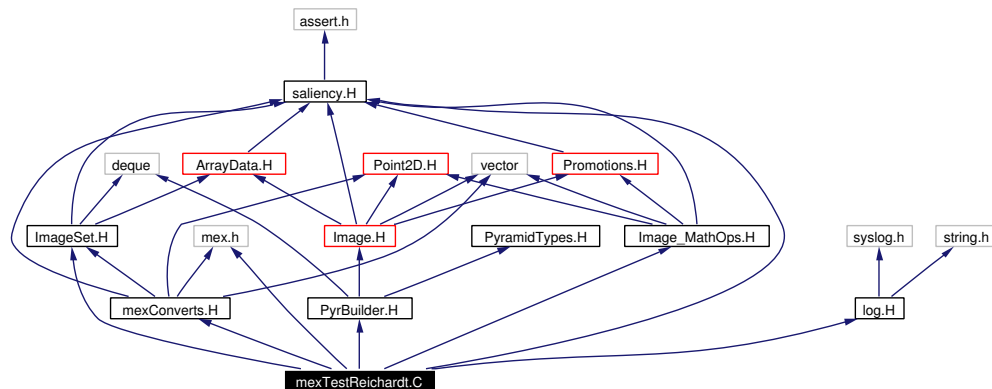
Definition in file [mexTestDirectFeedChannel.C](#).



## 11.248 mexTestReichardt.C File Reference

```
#include <mex.h>
#include "saliency.H"
#include "Image_MathOps.H"
#include "log.H"
#include "ImageSet.H"
#include "PyrBuilder.H"
#include "mexConverts.H"
```

Include dependency graph for mexTestReichardt.C:



### Functions

- void **mexFunction** (int nlhs, mxArray \*plhs[ ], int nrhs, const mxArray \*prhs[ ])

#### 11.248.1 Detailed Description

Test the Reichardt Pyramid (get tuning curves)

This file compiles to TestReichardt.mexglx which can be used to derive tuning curves for the ReichardtPyramid.

Call:

```
[avg, min, max, levMap] = TestReichardt(img_seq, [dx dy],
depth, lev)
```

Input Arguments:

**Parameters:**

***img\_seq*** a 3d array, containing a series of images

***[dx dy]*** displacement determining the direction and magnitude of motion to be tested

***depth*** depth of the ReichardtPyramid

***lev*** level for which the Reichardt maps are returned

Return Values:

**Parameters:**

***avg*** 2d array containing the avg activation of each level of the ReichardtPyramid for each input image

***min*** the same, only this time it's the minimum

***max*** the same with the maximum

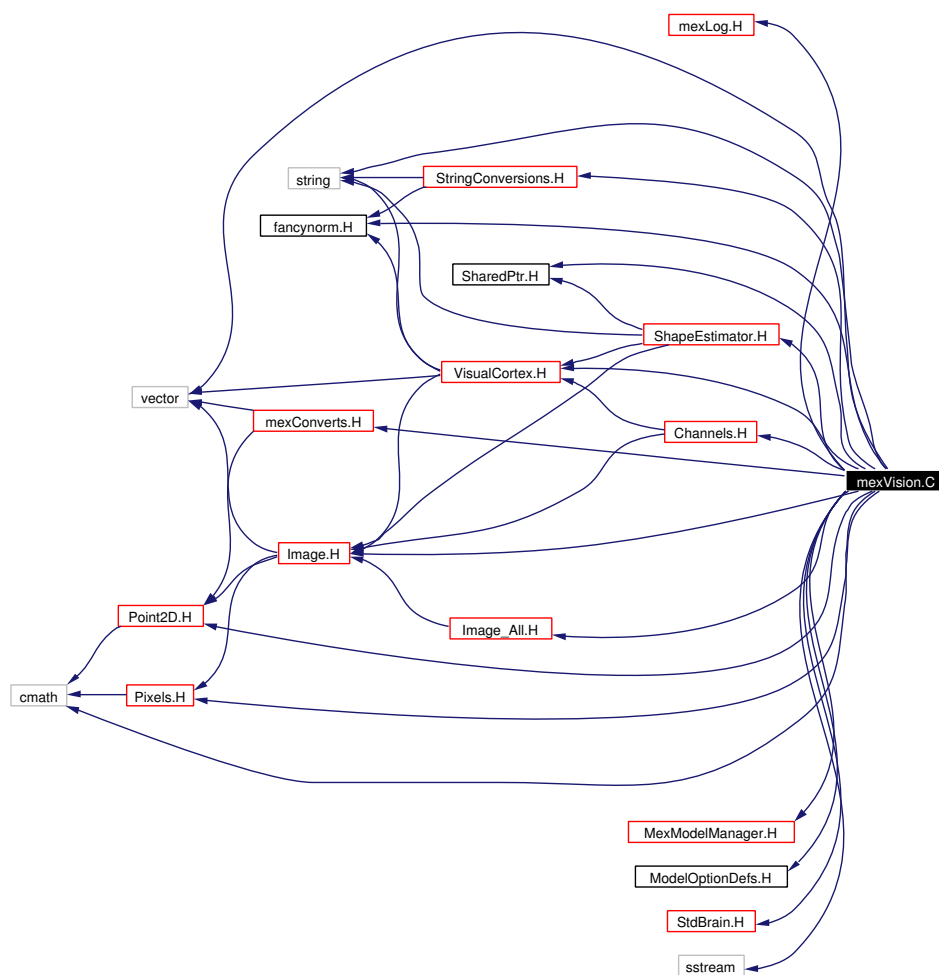
***levMap*** 3d array containing the 'lev'th Reichardt map for each frame

Definition in file [mexTestReichardt.C](#).

## 11.249 mexVision.C File Reference

```
#include "mexLog.H"
#include "mexConverts.H"
#include "Channels.H"
#include "MexModelManager.H"
#include "ModelOptionDefs.H"
#include "SharedPtr.H"
#include "StdBrain.H"
#include "VisualCortex.H"
#include "ShapeEstimator.H"
#include "fancynorm.H"
#include "Image.H"
#include "Point2D.H"
#include "Image_All.H"
#include "Pixels.H"
#include "StringConversions.H"
#include <vector>
#include <string>
#include <cmath>
#include <sstream>
```

Include dependency graph for mexVision.C:



## Functions

- void **mexFunction** (int nlhs, mxArray \*plhs[ ], int nrhs, const mxArray \*prhs[ ])

### 11.249.1 Detailed Description

compute saliency for an image

```

////////////////////////////////////// // The iLab Neuromorphic Vision C++
Toolkit - Copyright (C) 2001 by the // University of Southern California (USC)

```

and the iLab at USC. // See <http://iLab.usc.edu> for information about this project. //   
 // Major portions of the iLab Neuromorphic Vision Toolkit are protected // under the U.S. patent “Computation of Intrinsic Perceptual Saliency // in Visual Environments, and Applications” by Christof Koch and // Laurent Itti, California Institute of Technology, 2001 (patent // pending; application number 09/912,225 filed July 23, 2001; see // <http://pair.uspto.gov/cgi-bin/final/home.pl> for current status). //   
 // This file is part of the iLab Neuromorphic Vision C++ Toolkit. // // The iLab Neuromorphic Vision C++ Toolkit is free software; you can // redistribute it and/or modify it under the terms of the GNU General // Public License as published by the Free Software Foundation; either // version 2 of the License, or (at your option) any later version. // // The iLab Neuromorphic Vision C++ Toolkit is distributed in the hope // that it will be useful, but WITHOUT ANY WARRANTY; without even the // implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR // PURPOSE. See the GNU General Public License for more details. // // You should have received a copy of the GNU General Public License // along with the iLab Neuromorphic Vision C++ Toolkit; if not, write // to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, // Boston, MA 02111-1307 USA. //   
 //

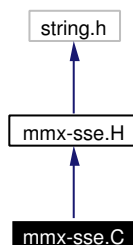
Primary maintainer for this file: Dirk Walther <[walther@caltech.edu](mailto:walther@caltech.edu)>

Definition in file [mexVision.C](#).

## 11.250 mmx-sse.C File Reference

```
#include "mmx-sse.H"
```

Include dependency graph for mmx-sse.C:



### Typedefs

- typedef int **int32**
- typedef unsigned char **byte**
- typedef float **float32**

### 11.250.1 Detailed Description

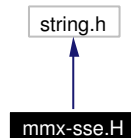
– Optimized implementations of low-level functions for MMX/SSE

Definition in file [mmx-sse.C](#).

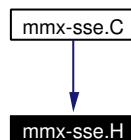
## 11.251 mmx-sse.H File Reference

```
#include <string.h>
```

Include dependency graph for mmx-sse.H:



This graph shows which files directly or indirectly include this file:



### Typedefs

- typedef int **int32**
- typedef unsigned char **byte**
- typedef float **float32**

### Functions

- void [sse\\_lowPass9x](#) (const float \*, float \*, int, int)

*In all the functions.*

#### 11.251.1 Detailed Description

Optimized implementations of low-level functions for MMX/SSE

Definition in file [mmx-sse.H](#).

## 11.251.2 Function Documentation

### 11.251.2.1 void sse\_lowPass9x (const float \*, float \*, int, int)

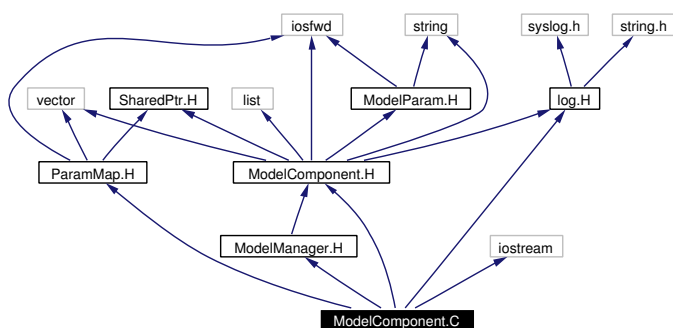
In all the functions.



## 11.252 ModelComponent.C File Reference

```
#include "ModelComponent.H"  
#include "ModelManager.H"  
#include "ParamMap.H"  
#include "log.H"  
#include <iostream>
```

Include dependency graph for ModelComponent.C:



### 11.252.1 Detailed Description

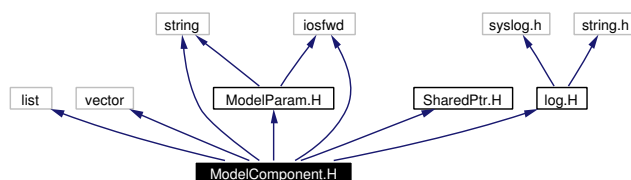
base class for parameterized model components

Definition in file [ModelComponent.C](#).

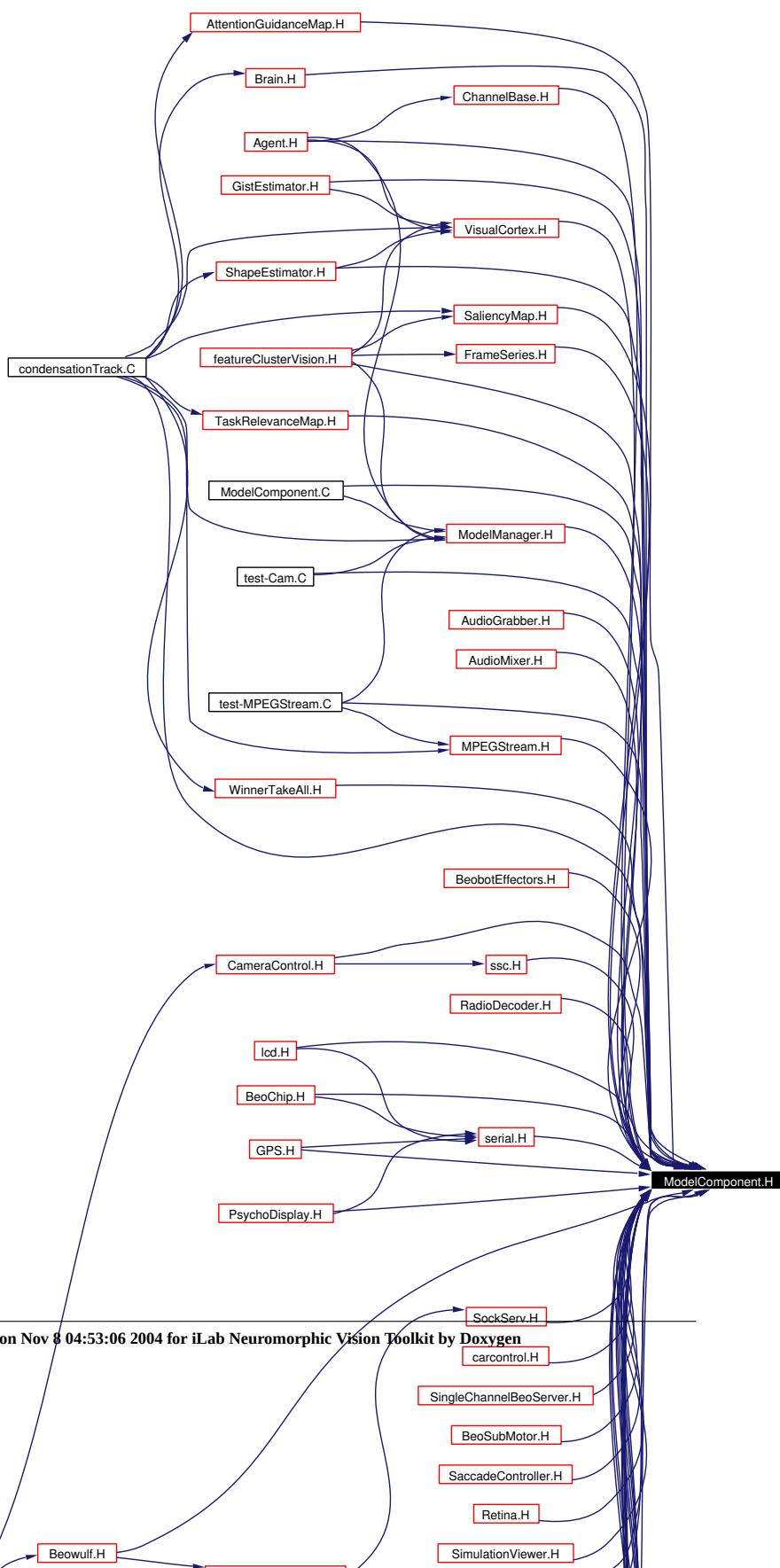
## 11.253 ModelComponent.H File Reference

```
#include <list>
#include <vector>
#include <string>
#include <iosfwd>
#include "ModelParam.H"
#include "SharedPtr.H"
#include "log.H"
```

Include dependency graph for ModelComponent.H:



This graph shows which files directly or indirectly include this file:



## Classes

- class [ModelComponent](#)

*Base class for parameterized model components.*

### 11.253.1 Detailed Description

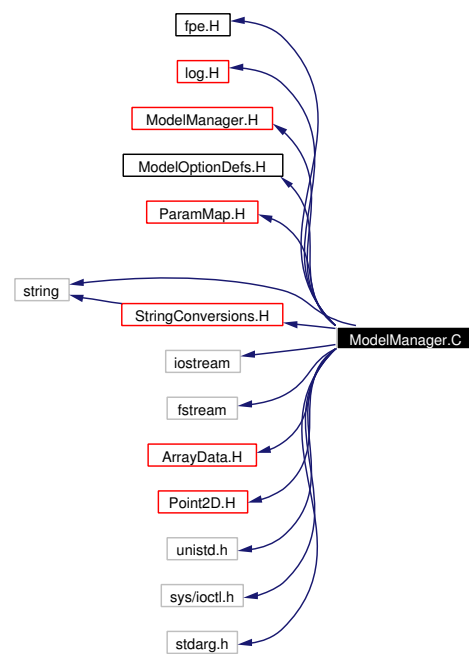
base class for parameterized model components

Definition in file [ModelComponent.H](#).

## 11.254 ModelManager.C File Reference

```
#include "fpe.H"
#include "log.H"
#include "ModelManager.H"
#include "ModelOptionDefs.H"
#include "ParamMap.H"
#include "StringConversions.H"
#include <iostream>
#include <string>
#include <fstream>
#include "ArrayData.H"
#include "Point2D.H"
#include <unistd.h>
#include <sys/ioctl.h>
#include <stdarg.h>
```

Include dependency graph for ModelManager.C:



### 11.254.1 Detailed Description

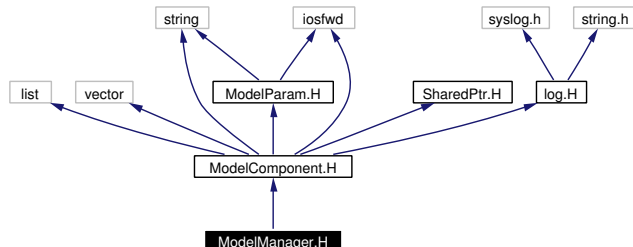
manages a model as collection of `ModelComponents`

Definition in file [ModelManager.C](#).

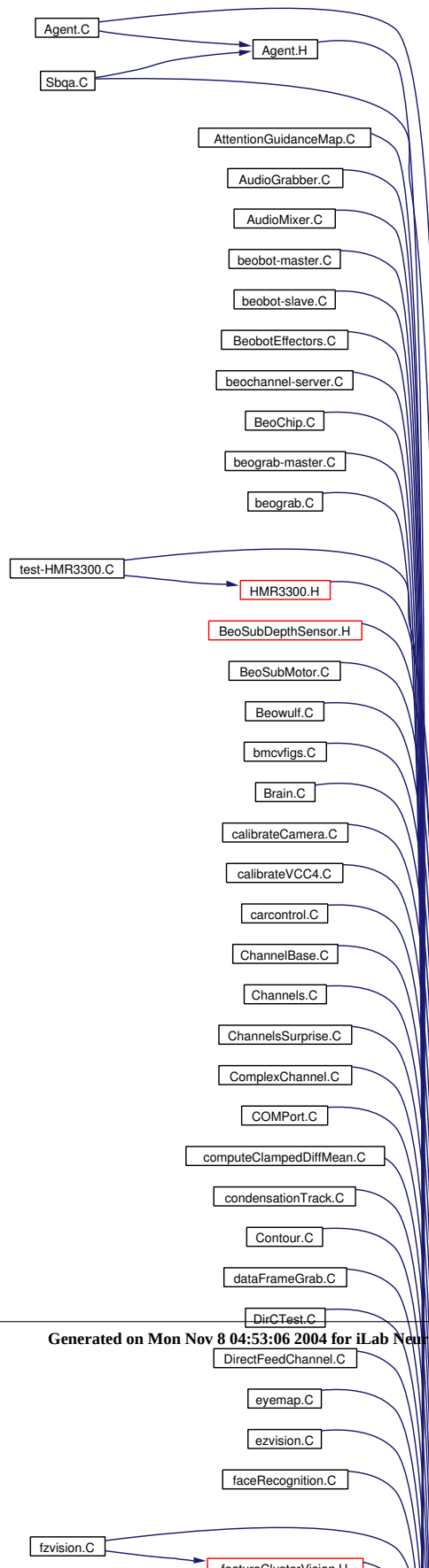
## 11.255 ModelManager.H File Reference

```
#include "ModelComponent.H"
```

Include dependency graph for ModelManager.H:



This graph shows which files directly or indirectly include this file:





## Classes

- class [ModelManager](#)  
*Manages a collection of [ModelComponent](#) objects forming a model.*
- struct **ModelManager::ModelOption**

### 11.255.1 Detailed Description

manages a model as collection of ModelComponents

Definition in file [ModelManager.H](#).

## 11.256 ModelManagerControl.ui.h File Reference

### 11.256.1 Detailed Description

functions relating to model control panel

Definition in file [ModelManagerControl.ui.h](#).

## 11.257 ModelManagerDialog.ui.h File Reference

### Functions

- char \*\* [argv2qt](#) (int argc, const char \*\*argv)  
*satisfy Qt's annoying request for an argv vector that is NOT const*

### 11.257.1 Detailed Description

functions relating to model configuration dialog

Definition in file [ModelManagerDialog.ui.h](#).

### 11.257.2 Function Documentation

#### 11.257.2.1 char\*\* argv2qt (int *argc*, const char \*\* *argv*)

satisfy Qt's annoying request for an argv vector that is NOT const

Definition at line 102 of file ModelManagerDialog.ui.h.

## 11.258 ModelOptionAliases.H File Reference

This graph shows which files directly or indirectly include this file:



### Defines

- `#define REQUEST_OPTIONALIAS_ATTENTION(m)`  
*Request option aliases usable by an attention model.*

### 11.258.1 Detailed Description

Macros for instantiation of command-line option aliases

Definition in file [ModelOptionAliases.H](#).

### 11.258.2 Define Documentation

#### 11.258.2.1 `#define REQUEST_OPTIONALIAS_ATTENTION(m)`

##### Value:

```
m.requestOptionAlias("ALIASmovie"); \
m.requestOptionAlias("ALIASmoviefov"); \
m.requestOptionAlias("ALIASmovieeyehead"); \
m.requestOptionAlias("ALIASsurprise");
```

Request option aliases usable by an attention model.

Argument should be a [ModelManager](#) object

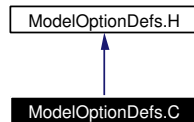
Definition at line 47 of file [ModelOptionAliases.H](#).

Referenced by [main\(\)](#).

## 11.259 ModelOptionDefs.C File Reference

```
#include "ModelOptionDefs.H"
```

Include dependency graph for ModelOptionDefs.C:



### Variables

- `const char * AllModelOptionCategs [NBMODELOPTIONCATEGS]`  
*text description of all option categories*
- `const ModelOptionDef AllModelOptions [ ]`  
*Our static array of all known model options, defined in [ModelOptionDefs.C](#).*

### 11.259.1 Detailed Description

Global list of all known ModelOption instances

Definition in file [ModelOptionDefs.C](#).

### 11.259.2 Variable Documentation

#### 11.259.2.1 `const char* AllModelOptionCategs[NBMODELOPTIONCATEGS]`

**Initial value:**

```
{
  "General Simulation Options",
  "Input Reading/Formatting Options",
  "Output Writing/Formatting Options",
  "What to Display/Save Options",
  "Audio-Related Options",
  "Retina-Related ooptions",
  "Channel-Related Options",
  "General Brain/Attention-Related Options",
  "Eye/Head Saccade Controller Options",
  "Video FrameGrabber-Related Options",
  "Beowulf-Related Options",
}
```

```
"Special-Hardware-Related Options",  
"PsychoDisplay-Related Options",  
"MBARI Result Viewer Related Options",  
"ImageDisplay Related Options",  
"Saliency Map Related Options",  
"Task-Relevance Map Related Options",  
"Attention Guidance Map Related Options",  
"Winner-Take-All Related Options",  
"Visual Buffer Related Options",  
"I-Room Related Options",  
"Vision Cluster Feature Related Options",  
"Inferior Temporal Cortex Options",  
"Gist Estimator Options",  
  
"Option Aliases and Shortcuts (may not always work)"  
}
```

text description of all option categories

Definition at line 40 of file ModelOptionDefs.C.

Referenced by ModelManager::printHelp().

#### 11.259.2.2 `const ModelOptionDef AllModelOptions[]`

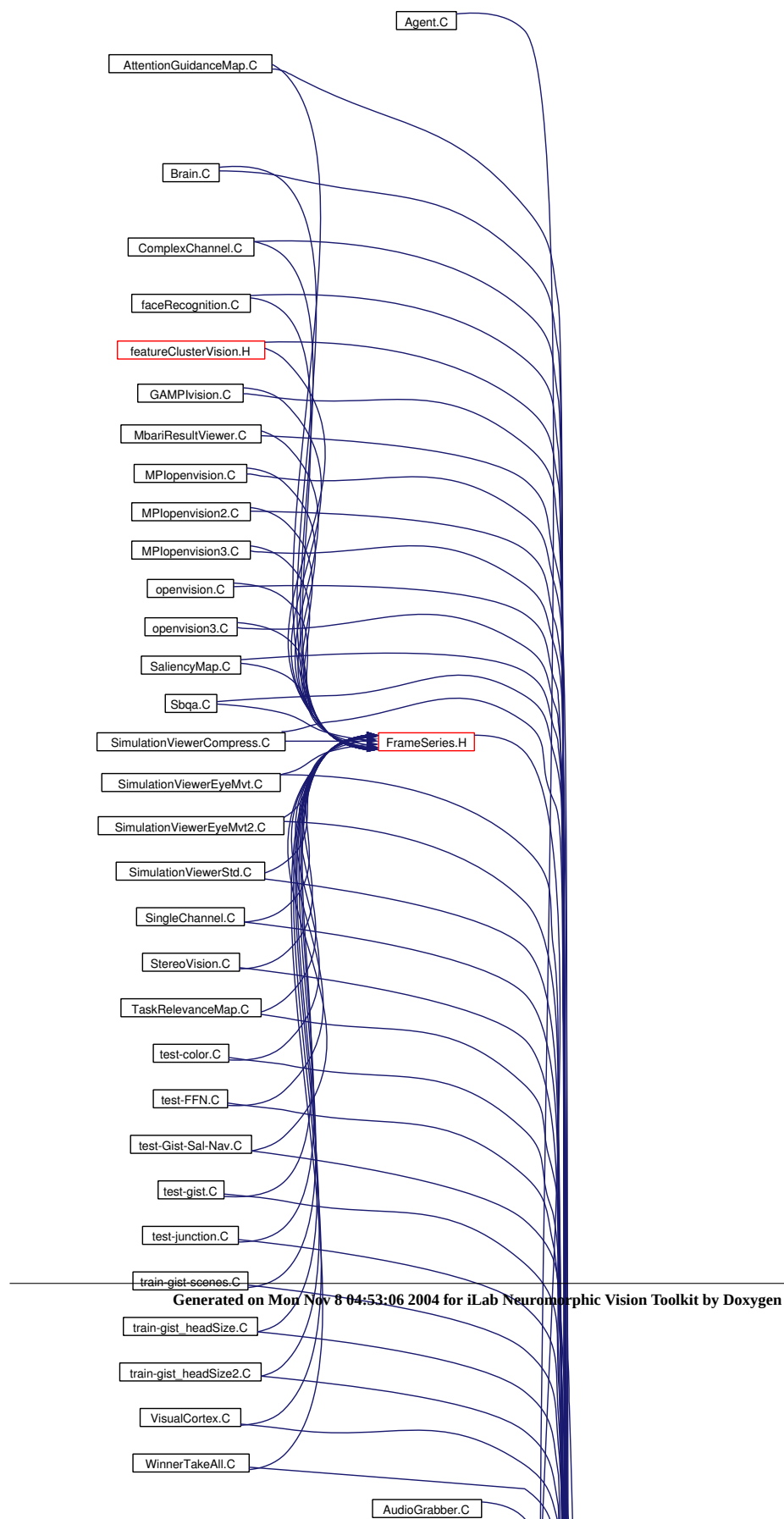
Our static array of all known model options, defined in [ModelOptionDefs.C](#).

Definition at line 89 of file ModelOptionDefs.C.

Referenced by ModelManager::ModelManager(), ModelManager::printHelp(), and ModelManager::setOptionValString().

## 11.260 ModelOptionDefs.H File Reference

This graph shows which files directly or indirectly include this file:





## Classes

- struct [ModelOptionDef](#)  
*Specification of a known model option.*

## Defines

- #define [NBMODELOPTIONCATEGS](#) 25  
*number of option categories*
- #define [OPTEXP\\_NONE](#) 0  
*What to export when calling [ModelComponent::exportOptions\(\)](#) Export no option.*
- #define [OPTEXP\\_ALL](#) 0xffff  
*Export all options.*
- #define [OPTEXP\\_CORE](#) (1 << 0)
- #define [OPTEXP\\_SAVE](#) (1 << 1)
- #define [OPTEXP\\_MRV](#) (1 << 2)

## Enumerations

- enum [ModelOptionType](#) {  
**MODEOPT\_NONE, MODEOPT\_BOOL, MODEOPT\_INT, MODEOPT\_UINT,**  
**MODEOPT\_FLOAT,      MODEOPT\_DOUBLE,      MODEOPT\_STRING,**  
**MODEOPT\_DIMS,**  
**MODEOPT\_POINT2D,    MODEOPT\_FLAG,    MODEOPT\_GRABMODE,**  
**MODEOPT\_FRAMERANGE,**  
**MODEOPT\_LEVELSPEC,    MODEOPT\_RECTANGLE,    MODEOPT\_-**  
**RASTERFFORMAT, MODEOPT\_RGB,**  
**MODEOPT\_PTYPE,    MODEOPT\_MAXNORM,    MODEOPT\_SEMODE,**  
**MODEOPT\_SESMOOTHMETHOD,**  
**MODEOPT\_INADDR, MODEOPT\_SHORT, MODEOPT\_ORICOMPTYPE,**  
**MODEOPT\_IORTYPE,**  
**MODEOPT\_BODRAWMODE,      MODEOPT\_IMAGEDISPLAYTYPE,**  
**MODEOPT\_ALIAS, MODEOPT\_COLCOMPTYPE }**  
*Known model option types.*

- enum [ModelOptionCateg](#) {  
[MOC\\_GENERAL](#) = 0, [MOC\\_INPUT](#) = 1, [MOC\\_OUTPUT](#) = 2, [MOC\\_DISPLAY](#) = 3,  
[MOC\\_AUDIO](#) = 4, [MOC\\_RETINA](#) = 5, [MOC\\_CHANNEL](#) = 6, [MOC\\_BRAIN](#) = 7,  
[MOC\\_SAC](#) = 8, [MOC\\_FRAMEGRAB](#) = 9, [MOC\\_BEOWULF](#) = 10, [MOC\\_SPECHARD](#) = 11,  
[MOC\\_PSYCHODISP](#) = 12, [MOC\\_MBARIRV](#) = 13, [MOC\\_IMAGEDISP](#) = 14, [MOC\\_SM](#) = 15,  
[MOC\\_TRM](#) = 16, [MOC\\_AGM](#) = 17, [MOC\\_WTA](#) = 18, [MOC\\_VB](#) = 19,  
[MOC\\_IROOM](#) = 20, [MOC\\_FCV](#) = 21, [MOC\\_ITC](#) = 22, [MOC\\_GE](#) = 23,  
[MOC\\_ALIAS](#) = 24 }

*Known model option categories; used for printing –help message.*

## Variables

- const char \* [AllModelOptionCategs](#) [NBMODELOPTIONCATEGS]  
*text description of all option categories*
- const [ModelOptionDef](#) [AllModelOptions](#) [ ]  
*Our static array of all known model options, defined in [ModelOptionDefs.C](#).*

### 11.260.1 Detailed Description

Specification for declaration of all valid model options

Definition in file [ModelOptionDefs.H](#).

### 11.260.2 Define Documentation

#### 11.260.2.1 #define NBMODELOPTIONCATEGS 25

number of option categories

Definition at line 91 of file [ModelOptionDefs.H](#).

Referenced by [ModelManager::printHelp\(\)](#).

**11.260.2.2 #define OPTEXP\_ALL 0xffff**

Export all options.

Definition at line 122 of file ModelOptionDefs.H.

Referenced by `main()`, `ModelManager::parseCommandLine()`, `MexModelManager::parseCommandLine()`, `VisualCortexEyeMvt::start1()`, and `V4LgrabberLight::V4LgrabberLight()`.

**11.260.2.3 #define OPTEXP\_CORE (1 << 0)**

Export our core configuration options that do not require any other model component to be present in the model

Definition at line 126 of file ModelOptionDefs.H.

Referenced by `WinnerTakeAllConfigurator::exportOptions()`, `WinnerTakeAll::exportOptions()`, `VisualCortexEyeMvt::exportOptions()`, `VisualCortexConfigurator::exportOptions()`, `VisualCortex::exportOptions()`, `VisualBuffer::exportOptions()`, `VCC4::exportOptions()`, `TCPcommunicator::exportOptions()`, `TaskRelevanceMapKillN::exportOptions()`, `TaskRelevanceMapKillStatic::exportOptions()`, `TaskRelevanceMapStd::exportOptions()`, `TaskRelevanceMapConfigurator::exportOptions()`, `TaskRelevanceMap::exportOptions()`, `SoxChannel::exportOptions()`, `SockServ::exportOptions()`, `SingleChannelSurprise::exportOptions()`, `SingleChannelBeoServer::exportOptions()`, `SingleChannel::exportOptions()`, `SimulationViewerStd::exportOptions()`, `SimulationViewerEyeMvt2::exportOptions()`, `SimulationViewerEyeMvt::exportOptions()`, `SimulationViewerCompress::exportOptions()`, `SimulationViewerConfigurator::exportOptions()`, `ShapeEstimator::exportOptions()`, `SaliencyMapFast::exportOptions()`, `SaliencyMapStd::exportOptions()`, `SaliencyMapConfigurator::exportOptions()`, `SaliencyMap::exportOptions()`, `ModelEyeHeadController::exportOptions()`, `ModelWrapperSaccadeController::exportOptions()`, `VisBufSaccadeController::exportOptions()`, `EyeTrackerSaccadeController::exportOptions()`, `MonkeySaccadeController::exportOptions()`, `Monkey2SaccadeController::exportOptions()`, `MonkeyEyeHeadController::exportOptions()`, `ThresholdFrictionSaccadeController::exportOptions()`, `ThresholdSaccadeController::exportOptions()`, `FrictionSaccadeController::exportOptions()`, `FixedSaccadeController::exportOptions()`, `SaccadeControllerConfigurator::exportOptions()`, `RetinaStd::exportOptions()`, `RetinaConfigurator::exportOptions()`, `RadioDecoder::exportOptions()`, `PsychoDisplay::exportOptions()`, `MPEGStream::exportOptions()`, `InferoTemporalConfigurator::exportOptions()`, `InferoTemporal::exportOptions()`, `ImageDisplay::exportOptions()`, `GistEstimatorConfigurator::exportOptions()`, `FrameSeries::exportOptions()`, `FrameGrabberConfigurator::exportOptions()`, `FrameGrabber::exportOptions()`, `DirectFeedChannel::exportOptions()`, `ComplexChannel::exportOptions()`, `DummyChannel::exportOptions()`, `ScorrChannel::exportOptions()`, `TcorrChannel::exportOptions()`, `StereoChannel::exportOptions()`, `EndPointChannel::exportOptions()`, `XJunctionChannel::export-`

Options(), TJunctionChannel::exportOptions(), LJunctionChannel::exportOptions(), JunctionChannel::exportOptions(), MotionChannel::exportOptions(), DirectionChannel::exportOptions(), MultiColorBandChannel::exportOptions(), OrientationChannel::exportOptions(), ColorChannel::exportOptions(), GaborChannel::exportOptions(), ChannelBase::exportOptions(), Brain::exportOptions(), Beowulf::exportOptions(), AudioMixer::exportOptions(), AudioGrabber::exportOptions(), AttentionGuidanceMapConfigurator::exportOptions(), AttentionGuidanceMap::exportOptions(), and main().

#### 11.260.2.4 #define OPTEXP\_MRV (1 << 2)

Export options that relate to the [MbariResultViewer](#)

Definition at line 133 of file ModelOptionDefs.H.

Referenced by MbariResultViewer::exportOptions().

#### 11.260.2.5 #define OPTEXP\_NONE 0

What to export when calling [ModelComponent::exportOptions\(\)](#) Export no option.

Definition at line 119 of file ModelOptionDefs.H.

Referenced by AttentionGuidanceMapConfigurator::AttentionGuidanceMapConfigurator(), SoxChannel::buildSubChans(), MotionChannelSurprise::buildSubChans(), MultiColorBandChannelSurprise::buildSubChans(), OrientationChannelSurprise::buildSubChans(), MotionChannelBeo::buildSubChans(), MultiColorBandChannelBeo::buildSubChans(), OrientationChannelBeo::buildSubChans(), StereoChannel::buildSubChans(), MotionChannel::buildSubChans(), MultiColorBandChannel::buildSubChans(), OrientationChannel::buildSubChans(), EndPointChannel::EndPointChannel(), GistEstimatorConfigurator::GistEstimatorConfigurator(), InferoTemporalConfigurator::InferoTemporalConfigurator(), LJunctionChannel::LJunctionChannel(), main(), MotionChannel::MotionChannel(), MultiColorBandChannel::MultiColorBandChannel(), OrientationChannel::OrientationChannel(), RetinaConfigurator::RetinaConfigurator(), SaccadeControllerConfigurator::SaccadeControllerConfigurator(), SaliencyMapConfigurator::SaliencyMapConfigurator(), SimulationViewerCompress::SimulationViewerCompress(), SimulationViewerConfigurator::SimulationViewerConfigurator(), StereoChannel::StereoChannel(), TaskRelevanceMapConfigurator::TaskRelevanceMapConfigurator(), TJunctionChannel::TJunctionChannel(), VisualCortexConfigurator::VisualCortexConfigurator(), WinnerTakeAllConfigurator::WinnerTakeAllConfigurator(), and XJunctionChannel::XJunctionChannel().

### 11.260.2.6 `#define OPTEXP_SAVE (1 << 1)`

Export our options that relate to saving results, and require that an [OutputFrameSeries](#) be available (and used!) in the model

Definition at line 130 of file ModelOptionDefs.H.

Referenced by `WinnerTakeAll::exportOptions()`, `VisualCortex::exportOptions()`, `TaskRelevanceMap::exportOptions()`, `SingleChannel::exportOptions()`, `SaliencyMap::exportOptions()`, `ComplexChannel::exportOptions()`, `Brain::exportOptions()`, `AttentionGuidanceMap::exportOptions()`, and `main()`.

## 11.260.3 Enumeration Type Documentation

### 11.260.3.1 `enum ModelOptionCateg`

Known model option categories; used for printing `-help` message.

#### Enumeration values:

- `MOC_GENERAL`** General simulation options.
- `MOC_INPUT`** Input reading/formatting options.
- `MOC_OUTPUT`** Output writing/formatting options.
- `MOC_DISPLAY`** What to display/save options.
- `MOC_AUDIO`** Audio-related options.
- `MOC_RETINA`** Retina-related options.
- `MOC_CHANNEL`** Channel-related options.
- `MOC_BRAIN`** Brain-related options.
- `MOC_SAC`** SaccadeController-related options.
- `MOC_FRAMEGRAB`** FrameGrabber-related options.
- `MOC_BEOWULF`** Beowulf-related options.
- `MOC_SPECHARD`** Special-hardware options (e.g., pan/tilt cam, etc).
- `MOC_PSYCHODISP`** [PsychoDisplay](#) options.
- `MOC_MBARI`** [MbariResultViewer](#) options.
- `MOC_IMAGE`** [ImageDisplay](#) options.
- `MOC_SM`** [SaliencyMap](#) options.
- `MOC_TRM`** [TaskRelevanceMap](#) options.
- `MOC_AGM`** [AttentionGuidanceMap](#) options.
- `MOC_WTA`** [WinnerTakeAll](#) options.
- `MOC_VB`** [VisualBuffer](#) options.

***MOC\_IROOM*** I-Room related options.

***MOC\_FCV*** Feature Cluster Options.

***MOC\_ITC*** Inferior Temporal Cortex Options.

***MOC\_GE*** [GistEstimator](#) Options.

***MOC\_ALIAS*** Option aliases.

Definition at line 59 of file ModelOptionDefs.H.

#### 11.260.3.2 **enum** [ModelOptionType](#)

Known model option types.

If you add new types here, be sure to update [ModelParam.C](#), [StringConversions.\[CH\]](#) and [ModelManager.C](#) (function [ModelManager::parseCommandLine\(\)](#))

Definition at line 47 of file ModelOptionDefs.H.

### 11.260.4 **Variable Documentation**

#### 11.260.4.1 **const char\*** [AllModelOptionCategs](#)[NBMODELOPTIONCATEGS]

text description of all option categories

Definition at line 94 of file ModelOptionDefs.H.

Referenced by [ModelManager::printHelp\(\)](#).

#### 11.260.4.2 **const** [ModelOptionDef](#) [AllModelOptions](#)[]

Our static array of all known model options, defined in [ModelOptionDefs.C](#).

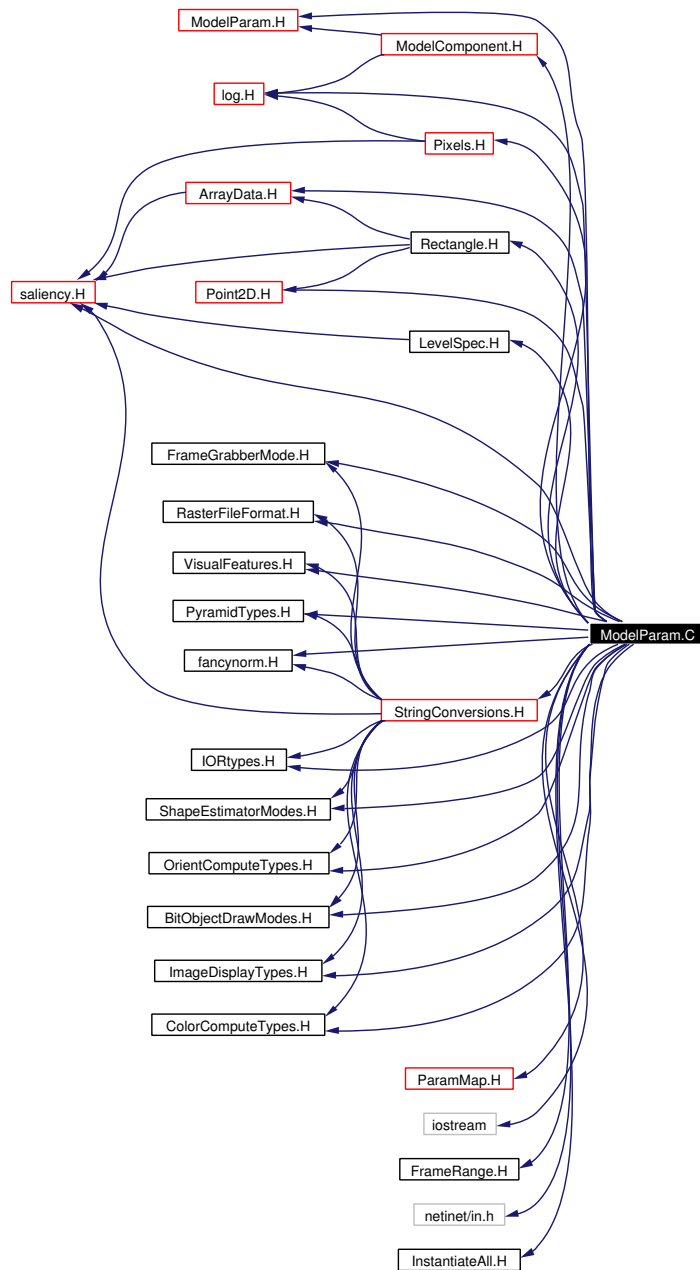
Definition at line 113 of file ModelOptionDefs.H.

Referenced by [ModelManager::ModelManager\(\)](#), [ModelManager::printHelp\(\)](#), and [ModelManager::setOptionValString\(\)](#).

## 11.261 ModelParam.C File Reference

```
#include "ModelParam.H"
#include "ParamMap.H"
#include "ModelComponent.H"
#include "log.H"
#include "StringConversions.H"
#include <iostream>
#include "saliency.H"
#include "ArrayData.H"
#include "Point2D.H"
#include "FrameGrabberMode.H"
#include "FrameRange.H"
#include "LevelSpec.H"
#include "Rectangle.H"
#include "RasterFileFormat.H"
#include "Pixels.H"
#include "VisualFeatures.H"
#include "PyramidTypes.H"
#include "fancynorm.H"
#include "IORtypes.H"
#include "ShapeEstimatorModes.H"
#include "OrientComputeTypes.H"
#include <netinet/in.h>
#include "BitObjectDrawModes.H"
#include "ImageDisplayTypes.H"
#include "ColorComputeTypes.H"
#include "InstantiateAll.H"
```

Include dependency graph for ModelParam.C:





## Defines

- #define `INstantiate(T)` template class `ModelParam< T >`;

### 11.261.1 Detailed Description

A tunable `ModelComponent` parameter base class

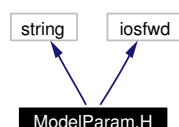
Definition in file `ModelParam.C`.

## 11.262 ModelParam.H File Reference

```
#include <string>
```

```
#include <iosfwd>
```

Include dependency graph for ModelParam.H:



This graph shows which files directly or indirectly include this file:

### Classes

- class [ModelParamBase](#)  
*ModelParamBase is a persistent [ModelComponent](#) parameter base class.*
- class [ModelParam](#)  
*ModelParam<T> definitions.*

### 11.262.1 Detailed Description

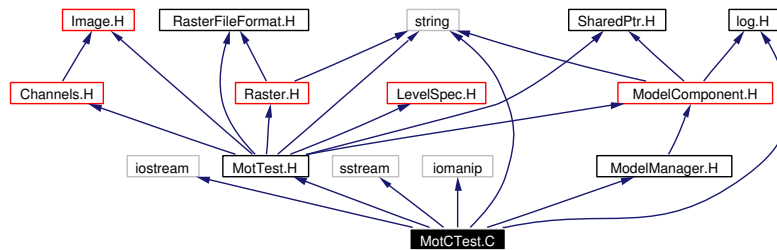
A tunable [ModelComponent](#) parameter base class

Definition in file [ModelParam.H](#).

## 11.263 MotCTest.C File Reference

```
#include <iostream>
#include <string>
#include <sstream>
#include <iomanip>
#include "MotTest.H"
#include "log.H"
#include "ModelManager.H"
```

Include dependency graph for MotCTest.C:



### Classes

- struct **Args**

### Defines

- `#define MYLEV 0`

### Functions

- `int main (int argc, char **argv)`

#### 11.263.1 Detailed Description

This program is for testing the [MotionChannel](#). It takes as input from the command line in the form:

```
MotCTest [OPTIONS] imageBaseName startFrame endFrame
```

Currently, most of the options aren't implemented in code. Instead, call `MotTest::display()` with the appropriate parameters for each map that need to be displayed.

This program uses the helper class [MotTest](#) in [MotTest.HC](#) to do most of the work.

Here are the planned options: (unimplemented)

The options determine which maps are displayed, whether the images are normalized by [XWinManaged](#) before they are displayed, and whether the [MotionChannel](#) should use intensity or gabor channels.

The valid options are: `-rs -rm -gps -gpm -ss -sm -cs -gm -dm -norm -ni`

`r` = Reichardt `gp` = Gabor pyramid `s` = Submap (feature) `c` = conspicuity (both from gabor and direction) `g` = gabor `d` = direction

The suffixes are:

`s` = statistics (min, max, avg) `m` = maps

Others:

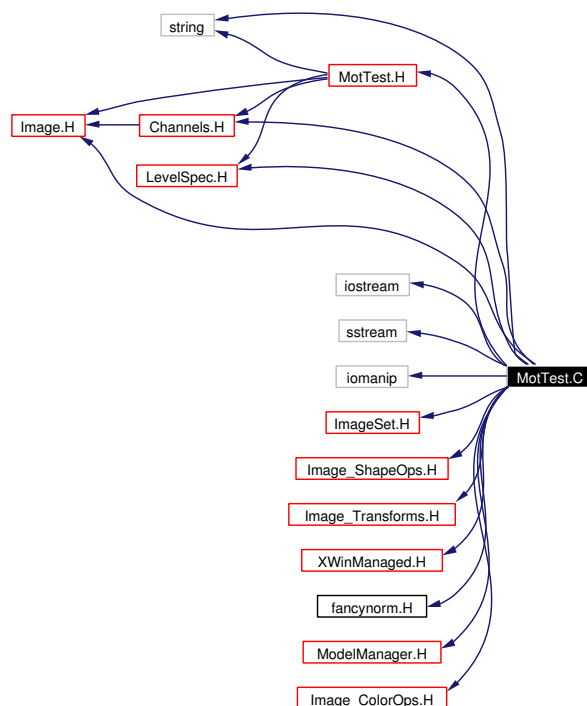
`norm` - normalize images `ni` - don't use intensity channel - use gabor instead

Definition in file [MotCTest.C](#).

## 11.264 MotTest.C File Reference

```
#include "MotTest.H"
#include <iostream>
#include <string>
#include <sstream>
#include <iomanip>
#include "Image.H"
#include "ImageSet.H"
#include "LevelSpec.H"
#include "Channels.H"
#include "Image_ShapeOps.H"
#include "Image_Transforms.H"
#include "XWinManaged.H"
#include "fancynorm.H"
#include "ModelManager.H"
#include "Image_ColorOps.H"
```

Include dependency graph for MotTest.C:



## Defines

- `#define Q_LEN 2`
- `#define SML 4`
- `#define DELTA_MIN 3`
- `#define DELTA_MAX 4`
- `#define LEVEL_MIN 2`
- `#define LEVEL_MAX 4`

### 11.264.1 Detailed Description

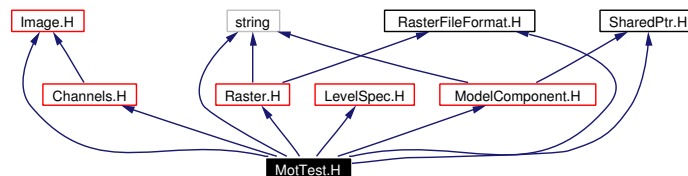
method implementations for class [MotTest](#)

Definition in file [MotTest.C](#).

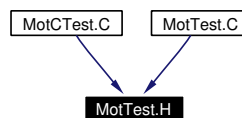
## 11.265 MotTest.H File Reference

```
#include <string>
#include "Image.H"
#include "Raster.H"
#include "LevelSpec.H"
#include "SharedPtr.H"
#include "RasterFileFormat.H"
#include "ModelComponent.H"
#include "Channels.H"
```

Include dependency graph for MotTest.H:



This graph shows which files directly or indirectly include this file:



### Classes

- struct **DispFlags**
- class [MotTest](#)  
Class for testing [MotionChannel](#).

### Defines

- #define **NUM\_FORM** 2
- #define **NUM\_CHAN\_TYPES** 3

- `#define NUM_CHAN 4`
- `#define NUM_GABOR 2`
- `#define NUM_DIR 4`
- `#define NUM_MOT 1`
- `#define NUM_MAP 3`
- `#define NUM_LEV 9`
- `#define FLAG_SIZE NUM_FORM * NUM_CHAN_TYPES * NUM_CHAN`  
    `* NUM_MAP`

### 11.265.1 Detailed Description

defines class [MotTest](#), used for testing the [MotionChannel](#) and [DirectionChannel](#) classes

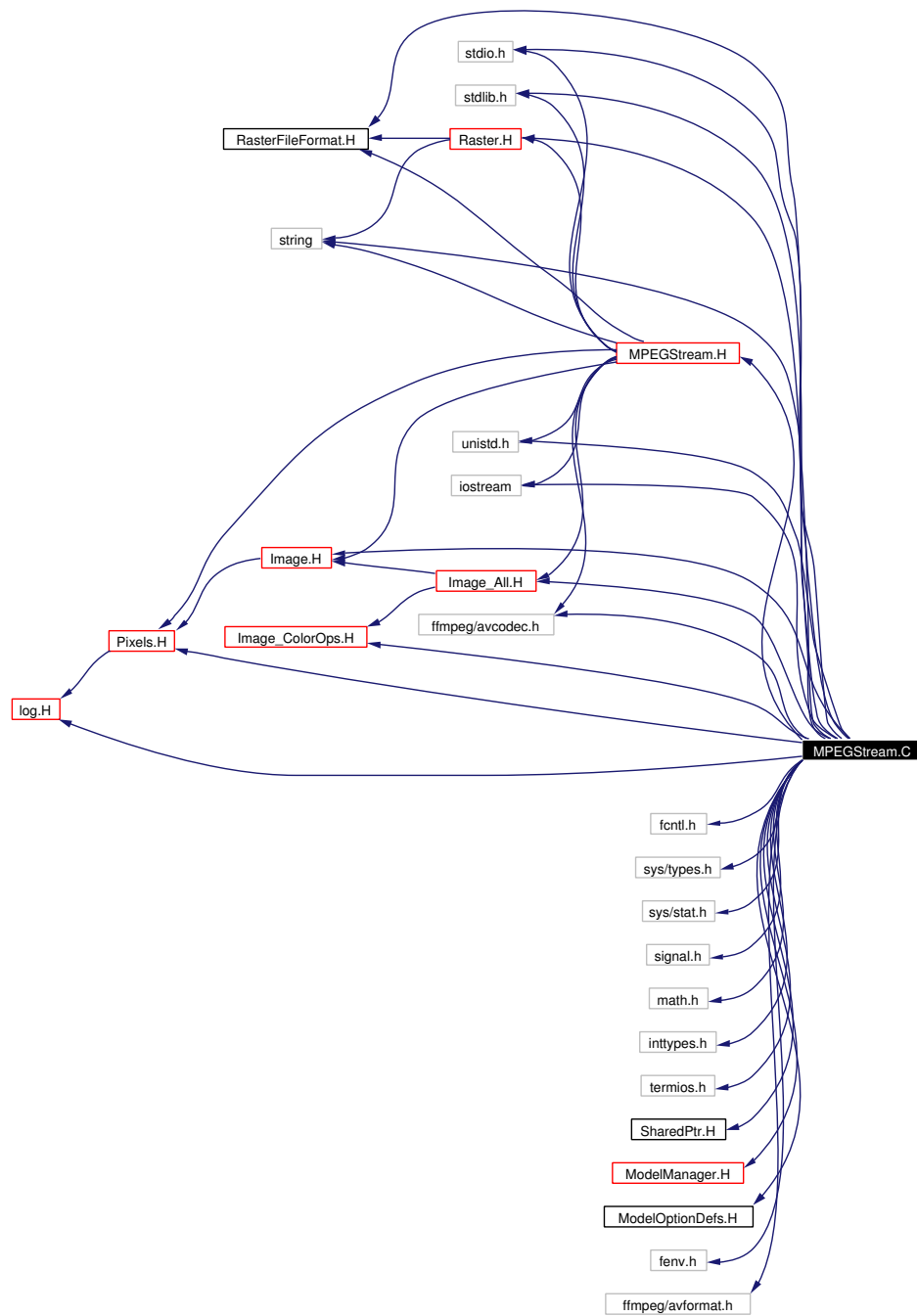
Definition in file [MotTest.H](#).



## 11.266 MPEGStream.C File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <string>
#include <unistd.h>
#include <fcntl.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <signal.h>
#include <iostream>
#include <math.h>
#include <inttypes.h>
#include <termios.h>
#include "SharedPtr.H"
#include "MPEGStream.H"
#include "Image.H"
#include "Image_All.H"
#include "RasterFileFormat.H"
#include "Raster.H"
#include "Pixels.H"
#include "Image_ColorOps.H"
#include "log.H"
#include "ModelManager.H"
#include "ModelOptionDefs.H"
#include <fenv.h>
#include <ffmpeg/avcodec.h>
#include <ffmpeg/avformat.h>
```

Include dependency graph for MPEGStream.C:



## Defines

- `#define MODE_RGB 0x1`
- `#define MODE_BGR 0x2`
- `#define MIN(A, B) ((A)<(B)?(A):(B))`
- `#define MAX(A, B) ((A)>(B)?(A):(B))`
- `#define Y_R_IN 0.257`
- `#define Y_G_IN 0.504`
- `#define Y_B_IN 0.098`
- `#define Y_ADD_IN 16`
- `#define U_R_IN 0.148`
- `#define U_G_IN 0.291`
- `#define U_B_IN 0.439`
- `#define U_ADD_IN 128`
- `#define V_R_IN 0.439`
- `#define V_G_IN 0.368`
- `#define V_B_IN 0.071`
- `#define V_ADD_IN 128`
- `#define SCALEBITS_IN 8`
- `#define FIX_IN(x) ((uint16_t) ((x) * (1L<<SCALEBITS_IN) + 0.5))`
- `#define RGB_Y_OUT 1.164`
- `#define B_U_OUT 2.018`
- `#define Y_ADD_OUT 16`
- `#define G_U_OUT 0.391`
- `#define G_V_OUT 0.813`
- `#define U_ADD_OUT 128`
- `#define R_V_OUT 1.596`
- `#define V_ADD_OUT 128`
- `#define SCALEBITS_OUT 13`
- `#define FIX_OUT(x) ((uint16_t) ((x) * (1L<<SCALEBITS_OUT) + 0.5))`

## Functions

- void **flip** (uint8\_t \*src, int width, int height)
- void **rgb24\_to\_yv12\_c** (uint8\_t \*y\_out, uint8\_t \*u\_out, uint8\_t \*v\_out, uint8\_t \*src, int width, int height, int stride)
- void **colorspace\_init** (void)
- void **yv12\_to\_rgb24\_c** (uint8\_t \*dst, int dst\_stride, uint8\_t \*y\_src, uint8\_t \*u\_src, uint8\_t \*v\_src, int y\_stride, int uv\_stride, int width, int height)

## Variables

- `int32_t RGB_Y_tab` [256]
- `int32_t B_U_tab` [256]
- `int32_t G_U_tab` [256]
- `int32_t G_V_tab` [256]
- `int32_t R_V_tab` [256]

### 11.266.1 Detailed Description

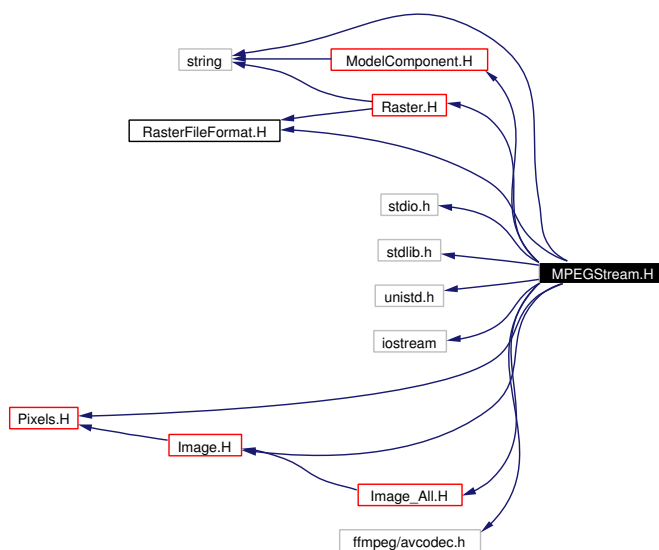
read frames from video streams

Definition in file [MPEGStream.C](#).

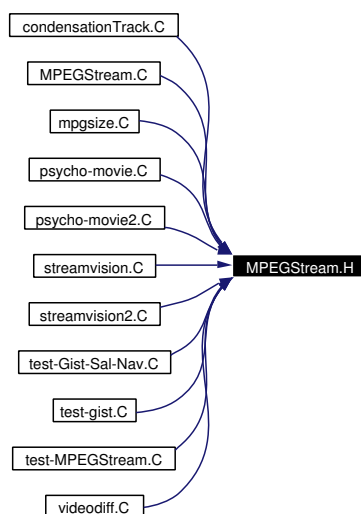
## 11.267 MPEGStream.H File Reference

```
#include "ModelComponent.H"
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <iostream>
#include <string>
#include "Image.H"
#include "Image_All.H"
#include "RasterFileFormat.H"
#include "Raster.H"
#include "Pixels.H"
#include <ffmpeg/avcodec.h>
```

Include dependency graph for MPEGStream.H:



This graph shows which files directly or indirectly include this file:



## Classes

- class [MPEGStream](#)

*MPEGStream is a wrapper class to read video frames from mpeg-1.*

## Defines

- `#define MPEG_DECODE 0`
- `#define MPEG_ENCODE 1`

### 11.267.1 Detailed Description

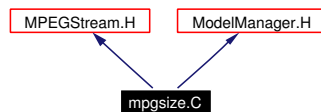
read frames from video streams

Definition in file [MPEGStream.H](#).

## 11.268 mpgsize.C File Reference

```
#include "MPEGStream.H"  
#include "ModelManager.H"
```

Include dependency graph for mpgsize.C:



### Functions

- int [main](#) (const int argc, const char \*\*argv)  
*Find size of an MPEG movie.*

#### 11.268.1 Detailed Description

find size of an MPEG movie

Definition in file [mpgsized.C](#).

#### 11.268.2 Function Documentation

##### 11.268.2.1 int main (const int argc, const char \*\* argv)

Find size of an MPEG movie.

This is basically like the standard `mpginfo` Linux command, but it also displays the number of frames in the stream. This is slow and inefficient as it reads and decodes the entire movie, just to finally get the number of frames...

Definition at line 47 of file `mpgsized.C`.

References `main()`, `MYLOGVERB`, and `uint`.

## 11.269 MPIopenvision.C File Reference

```
#include <iostream>
#include <fstream>
#include <cmath>
#include <mpi.h>
#include "saliency.H"
#include "ModelManager.H"
#include "Channels.H"
#include "VisualCortex.H"
#include "FrameSeries.H"
#include "ModelOptionDefs.H"
#include "Image_Transforms.H"
#include "Image_MathOps.H"
#include "log.H"
```

Include dependency graph for MPIopenvision.C:

### Defines

- #define **ITMAX** 200
- #define **FTOL** 0.00000001f
- #define **NBCOEFFS** 144
- #define **MASTER** 0
- #define **MSG\_DATA** 1000
- #define **MSG\_RESULT** 2000
- #define **PATH** "/tmphpc-01/itti/openvision"
- #define **NB\_PIC** 14
- #define **NB\_FILTERS** 3
- #define **NB\_EVALS** 1
- #define **NB\_PROC** (NBCOEFFS \* NB\_EVALS + 1)

### Functions

- int **main** (int argc, char \*\*argv)



### **11.269.1 Detailed Description**

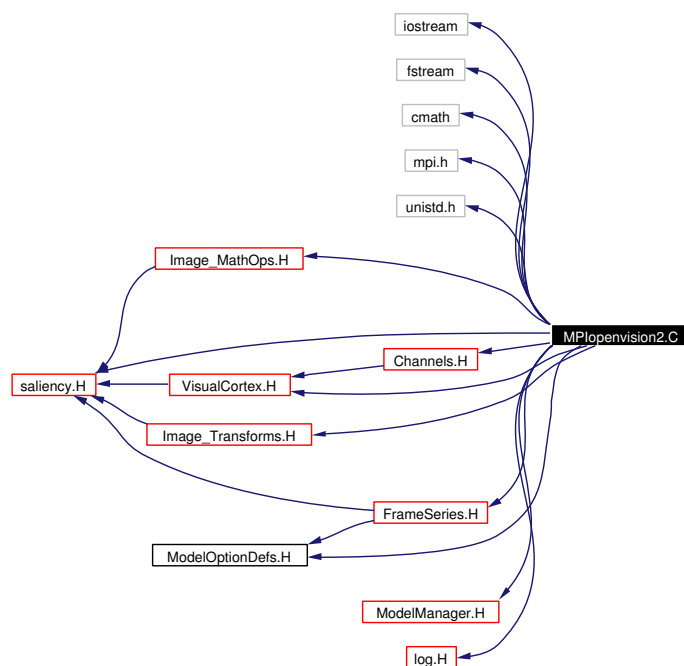
MPI version of [openvision.C](#)

Definition in file [MPIopenvision.C](#).

## 11.270 MPIopenvision2.C File Reference

```
#include <iostream>
#include <fstream>
#include <cmath>
#include <mpi.h>
#include <unistd.h>
#include "saliency.H"
#include "ModelManager.H"
#include "Channels.H"
#include "VisualCortex.H"
#include "FrameSeries.H"
#include "ModelOptionDefs.H"
#include "Image_Transforms.H"
#include "Image_MathOps.H"
#include "log.H"
```

Include dependency graph for MPIopenvision2.C:



## Defines

- #define **ITMAX** 200
- #define **FTOL** 0.000001f
- #define **MASTER** 0
- #define **MSG\_STOP** 10
- #define **MSG\_PIC** 1
- #define **MSG\_DATA** 2
- #define **MSG\_RESULT** 200
- #define **PATH** "/tmphpc-01/itti/openvision"
- #define **INPUTFILE** "/tmphpc-01/itti/openvision/start.txt"
- #define **NB\_PICS** 14
- #define **NB\_FILTERS** 3
- #define **NCOEFFS** 8
- #define **MCOEFFS** NCOEFFS \* NCOEFFS \* NB\_FILTERS \* 3
- #define **NB\_EVALS** 10
- #define **NB\_WAVES** 5
- #define **NB\_PROCS** NB\_PICS \* NB\_EVALS \* 2 + 1

## Functions

- int **main** (int argc, char \*\*argv)

### 11.270.1 Detailed Description

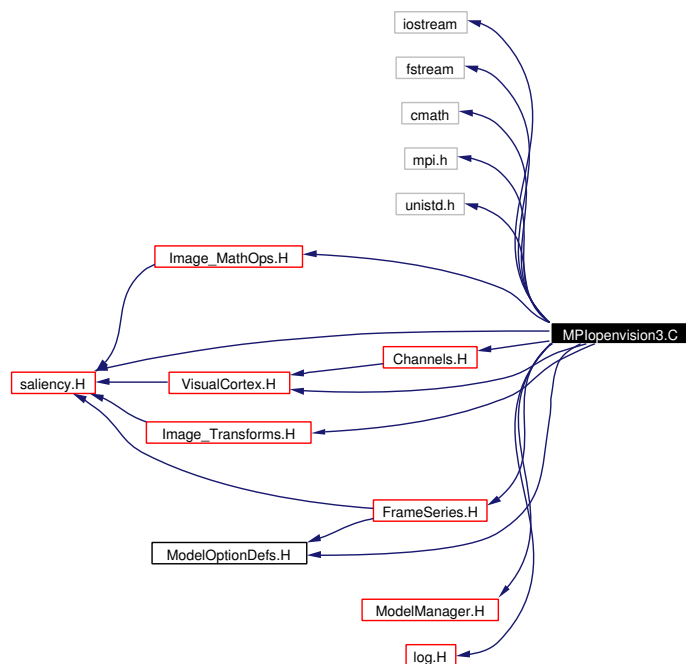
another MPI version of [openvision.C](#)

Definition in file [MPIopenvision2.C](#).

## 11.271 MPIopenvision3.C File Reference

```
#include <iostream>
#include <fstream>
#include <cmath>
#include <mpi.h>
#include <unistd.h>
#include "saliency.H"
#include "ModelManager.H"
#include "Channels.H"
#include "VisualCortex.H"
#include "FrameSeries.H"
#include "ModelOptionDefs.H"
#include "Image_Transforms.H"
#include "Image_MathOps.H"
#include "log.H"
```

Include dependency graph for MPIopenvision3.C:



## Defines

- #define **ITMAX** 200
- #define **FTOL** 0.000001f
- #define **MASTER** 0
- #define **MSG\_STOP** 10
- #define **MSG\_PIC** 1
- #define **MSG\_DATA** 2
- #define **MSG\_RESULT** 200
- #define **PATH** "/home/hpc-26/itti/openvision"
- #define **INPUTFILE** "/home/hpc-26/itti/openvision/start.txt"
- #define **NB\_PICS** 10
- #define **NB\_FILTERS** 3
- #define **NCOEFFS** 8
- #define **MCOEFFS** NCOEFFS \* NCOEFFS \* NB\_FILTERS
- #define **NB\_EVALS** 9
- #define **NB\_WAVES** 5
- #define **NB\_PROCS** NB\_PICS \* NB\_EVALS \* 2 + 1
- #define **RAD** 64

## Functions

- int **main** (int argc, char \*\*argv)

### 11.271.1 Detailed Description

MPI version of [openvision3.C](#)

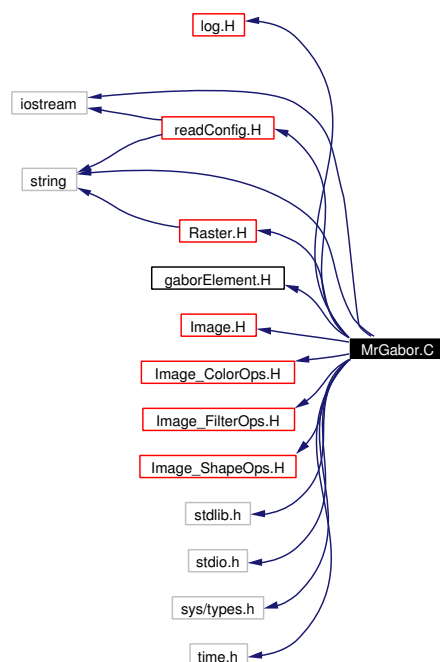
Definition in file [MPIopenvision3.C](#).

## 11.272 MrGabor.C File Reference

```
#include "log.H"  
#include "readConfig.H"  
#include "gaborElement.H"  
#include "Image.H"  
#include "Image_ColorOps.H"  
#include "Image_FilterOps.H"  
#include "Image_ShapeOps.H"  
#include "Raster.H"  
#include <stdlib.h>  
#include <stdio.h>  
#include <iostream>  
#include <string>  
#include <sys/types.h>  
#include <time.h>
```

Include dependency graph for MrGabor.C:





## Functions

- `int main (int argc, char *argv[ ])`

## Variables

- `char * configFile`  
*This is the configFile name.*
- `char * gaborElementFile`  
*This is the gaborElement file.*
- `const char * imageSaveName`  
*what to save the image as (name)*
- `readConfig configIn (25)`  
*This is the configFile object.*
- `readConfig gaborIn (25)`  
*This is the gaborElement object.*

- vector< gaborElement > [theGabors](#)  
*keep track of gabor properties*
- vector< [Image](#)< float > > [theGaborImages](#)  
*store the gabors*
- vector< [Image](#)< float > > [theMasterImages](#)  
*the master images*
- int [gaborNumber](#)  
*variables*
- int [sizeX](#)  
*variables*
- int [sizeY](#)  
*variables*
- float **backGround**
- char **getX** [100]
- char **getY** [100]
- char **getSTDDEV** [100]
- char **getPERIOD** [100]
- char **getPHASE** [100]
- char **getTHETA** [100]
- char **getSIGMOD** [100]
- char **getAMPLITUDE** [100]
- char **getMASTER** [100]

### 11.272.1 Detailed Description

A Gabor

Definition in file [MrGabor.C](#).

### 11.272.2 Variable Documentation

#### 11.272.2.1 char\* [configFile](#)

This is the configFile name.

Definition at line 29 of file MrGabor.C.

**11.272.2.2 readConfig configIn(25)**

This is the configFile object.

**11.272.2.3 char\* gaborElementFile**

This is the gaborElement file.

Definition at line 31 of file MrGabor.C.

**11.272.2.4 readConfig gaborIn(25)**

This is the gaborElement object.

**11.272.2.5 int gaborNumber**

variables

Definition at line 45 of file MrGabor.C.

**11.272.2.6 const char\* imageSaveName**

what to save the image as (name)

Definition at line 33 of file MrGabor.C.

**11.272.2.7 int sizeX**

variables

Definition at line 45 of file MrGabor.C.

Referenced by ContourConnection::ContourConnection(), and main().

**11.272.2.8 int sizeY**

variables

Definition at line 45 of file MrGabor.C.

Referenced by ContourConnection::ContourConnection(), and main().

**11.272.2.9** `vector<Image<float> >` [`theGaborImages`](#)

store the gabors

Definition at line 41 of file MrGabor.C.

**11.272.2.10** `vector<gaborElement>` [`theGabors`](#)

keep track of gabor properties

Definition at line 39 of file MrGabor.C.

**11.272.2.11** `vector<Image<float> >` [`theMasterImages`](#)

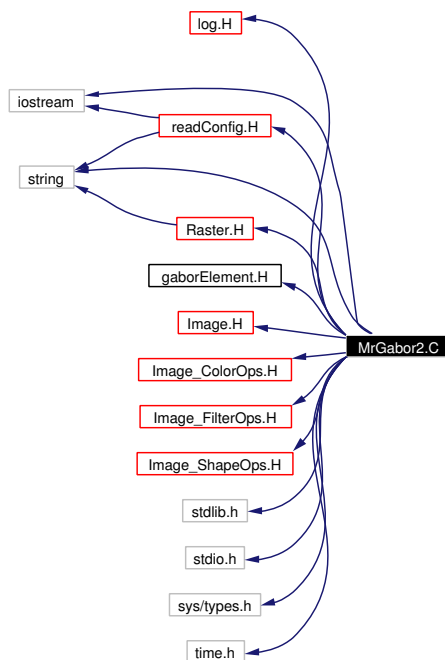
the master images

Definition at line 43 of file MrGabor.C.

## 11.273 MrGabor2.C File Reference

```
#include "log.H"
#include "readConfig.H"
#include "gaborElement.H"
#include "Image.H"
#include "Image_ColorOps.H"
#include "Image_FilterOps.H"
#include "Image_ShapeOps.H"
#include "Raster.H"
#include <stdlib.h>
#include <stdio.h>
#include <iostream>
#include <string>
#include <sys/types.h>
#include <time.h>
```

Include dependency graph for MrGabor2.C:



## Functions

- int **main** (int argc, char \*argv[ ])

## Variables

- char \* **configFile**  
*This is the configFile name.*
- char \* **gaborElementFile**  
*This is the gaborElement file.*
- const char \* **imageSaveName**  
*what to save the image as (name)*
- const char \* **saveDir**  
*what directory to save in*
- **readConfig** **configIn** (25)  
*This is the configFile object.*

- [readConfig gaborIn](#) (25)  
*This is the gaborElement object.*
- `vector< gaborElement >` [theGabors](#)  
*keep track of gabor properties*
- `vector< Image< float > >` [theGaborImages](#)  
*store the gabors*
- `vector< Image< float > >` [intermedImages](#)  
*store intermediate images*
- `vector< Image< float > >` [theMasterImages](#)  
*the master images*
- `int` [gaborNumber](#)  
*variables*
- `int` [sizeX](#)  
*variables*
- `int` [sizeY](#)  
*variables*
- `float` **backGround**
- `char` **getX** [100]
- `char` **getY** [100]
- `char` **getSTDDEV** [100]
- `char` **getPERIOD** [100]
- `char` **getPHASE** [100]
- `char` **getTHETA** [100]
- `char` **getSIGMOD** [100]
- `char` **getAMPLITUDE** [100]
- `char` **getMASTER** [100]

### 11.273.1 Detailed Description

This file is specifically designed to place gabors in line with polat and sagi 1994 and should not be used for generic gabor placement unless you understand this file -The Management

Definition in file [MrGabor2.C](#).

## 11.273.2 Variable Documentation

### 11.273.2.1 `char* configFile`

This is the configFile name.

Definition at line 31 of file MrGabor2.C.

### 11.273.2.2 `readConfig configIn(25)`

This is the configFile object.

### 11.273.2.3 `char* gaborElementFile`

This is the gaborElement file.

Definition at line 33 of file MrGabor2.C.

### 11.273.2.4 `readConfig gaborIn(25)`

This is the gaborElement object.

### 11.273.2.5 `int gaborNumber`

variables

Definition at line 51 of file MrGabor2.C.

### 11.273.2.6 `const char* imageSaveName`

what to save the image as (name)

Definition at line 35 of file MrGabor2.C.

### 11.273.2.7 `vector<Image<float> > intermedImages`

store intermediate images

Definition at line 47 of file MrGabor2.C.

### 11.273.2.8 `const char* saveDir`

what directory to save in



Definition at line 37 of file MrGabor2.C.

#### 11.273.2.9 int [sizeX](#)

variables

Definition at line 51 of file MrGabor2.C.

#### 11.273.2.10 int [sizeY](#)

variables

Definition at line 51 of file MrGabor2.C.

#### 11.273.2.11 vector<[Image](#)<float> > [theGaborImages](#)

store the gabors

Definition at line 45 of file MrGabor2.C.

#### 11.273.2.12 vector<[gaborElement](#)> [theGabors](#)

keep track of gabor properties

Definition at line 43 of file MrGabor2.C.

#### 11.273.2.13 vector<[Image](#)<float> > [theMasterImages](#)

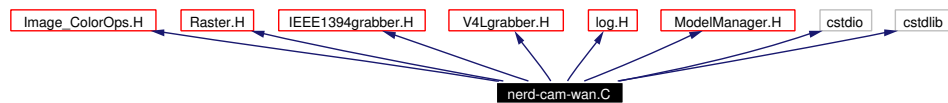
the master images

Definition at line 49 of file MrGabor2.C.

## 11.274 nerd-cam-wan.C File Reference

```
#include "Image_ColorOps.H"
#include "Raster.H"
#include "IEEE1394grabber.H"
#include "V4Lgrabber.H"
#include "log.H"
#include "ModelManager.H"
#include <stdio>
#include <stdlib>
```

Include dependency graph for nerd-cam-wan.C:



### Functions

- int [main](#) (const int argc, const char \*\*argv)  
*Does a single image grab and writes to file specified.*

### Variables

- const char \* [saveFile](#)

#### 11.274.1 Detailed Description

FIXME

Definition in file [nerd-cam-wan.C](#).

#### 11.274.2 Function Documentation

##### 11.274.2.1 int main (const int *argc*, const char \*\* *argv*)

Does a single image grab and writes to file specified.

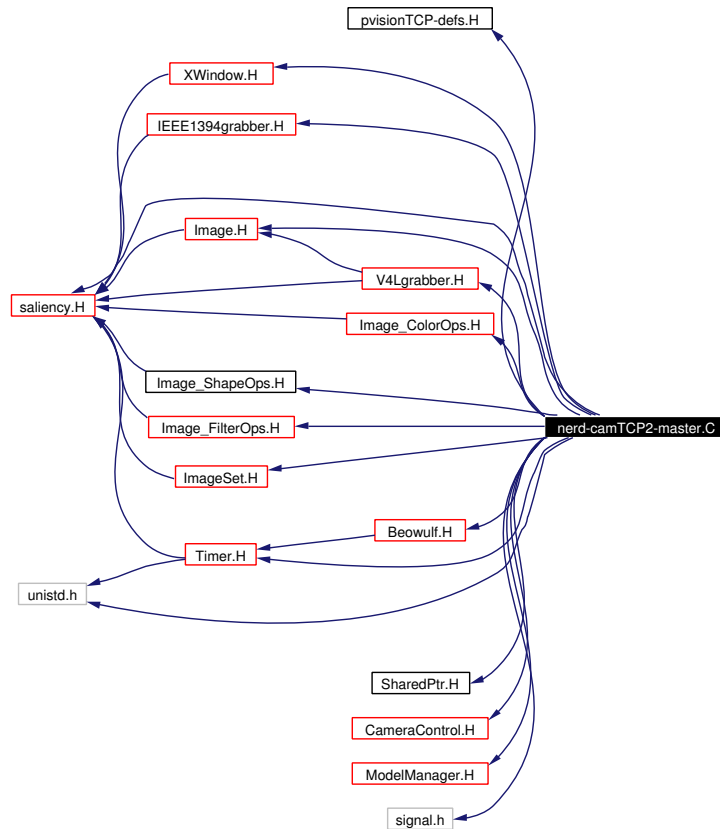
Definition at line 16 of file nerd-cam-wan.C.

References `SharedPtr< T >::get()`, `luminance()`, `main()`, `Raster::WriteGray()`, and `Raster::WriteRGB()`.

## 11.275    **nerd-camTCP2-master.C File Reference**

```
#include "pvisionTCP-defs.H"
#include "saliency.H"
#include "Beowulf.H"
#include "Image.H"
#include "Image_ColorOps.H"
#include "Image_ShapeOps.H"
#include "Image_FilterOps.H"
#include "ImageSet.H"
#include "XWindow.H"
#include "IEEE1394grabber.H"
#include "V4Lgrabber.H"
#include "SharedPtr.H"
#include "Timer.H"
#include "CameraControl.H"
#include "ModelManager.H"
#include <unistd.h>
#include <signal.h>
```

Include dependency graph for nerd-camTCP2-master.C:



## Defines

- #define **NAVG** 20  
*Number of frames over which average framerate is computed.*
- #define **NSTAT** 4  
*Number of stats collected.*
- #define **PRESCALE** 2  
*prescale level by which we downsize images before sending them off*
- #define **SLAVES** "n01 n02 n03 n04 n05 n06 n07 n08 n09 n10 n11 n12 n13 n14  
  \n15 n16"  
*Names of the beowulf slaves.*
- #define **SHIFTINT** 5

*shift attention at most once every SHIFTINT frames*

- `#define SMINJECT 0.6f`  
*injection ratio between new and older sm:*
- `#define SMTHRESH 200`  
*threshold salience (between 0 and 255) to warrant a shift of attention*

## Functions

- `void terminate (int s)`  
*Signal handler (e.g., for control-C).*
- `int main (const int argc, const char **argv)`

## Variables

- `bool goforever = true`  
*Will turn false on interrupt signal.*

### 11.275.1 Detailed Description

Grab & process over beowulf w/ pvisionTCP2

Definition in file [nerd-camTCP2-master.C](#).

### 11.275.2 Define Documentation

#### 11.275.2.1 `#define NAVG 20`

Number of frames over which average framerate is computed.

This parallel vision processing master is for use with pvisionTCP2. See the pvision-TCP2go script in bin/ for how to launch the slaves.

Definition at line 61 of file [nerd-camTCP2-master.C](#).

Referenced by `main()`.

**11.275.2.2 #define NSTAT 4**

Number of stats collected.

Definition at line 63 of file nerd-camTCP2-master.C.

**11.275.2.3 #define PRESCALE 2**

prescale level by which we downsize images before sending them off

Definition at line 66 of file nerd-camTCP2-master.C.

**11.275.2.4 #define SHIFTINT 5**

shift attention at most once every SHIFTINT frames

Definition at line 73 of file nerd-camTCP2-master.C.

**11.275.2.5 #define SLAVES "n01 n02 n03 n04 n05 n06 n07 n08 n09 n10 n11 n12  
n13 n14 \n15 n16"**

Names of the beowulf slaves.

Definition at line 69 of file nerd-camTCP2-master.C.

**11.275.2.6 #define SMINJECT 0.6f**

injection ratio between new and older sm:

Definition at line 76 of file nerd-camTCP2-master.C.

**11.275.2.7 #define SMTHRESH 200**

threshold salience (between 0 and 255) to warrant a shift of attention

Definition at line 79 of file nerd-camTCP2-master.C.

**11.275.3 Function Documentation****11.275.3.1 void terminate (int s)**

Signal handler (e.g., for control-C).

Definition at line 84 of file nerd-camTCP2-master.C.

References goforever.

## 11.275.4 Variable Documentation

### 11.275.4.1 `bool goforever = true` [static]

Will turn false on interrupt signal.

Definition at line 81 of file nerd-camTCP2-master.C.

Referenced by `terminate()`.



## 11.276 nerd-camTCP2.C File Reference

```
#include "pvisionTCP-defs.H"
#include "fancynorm.H"
#include "saliency.H"
#include "Beowulf.H"
#include "Pixels.H"
#include "Image.H"
#include "Image_ColorOps.H"
#include "Image_ShapeOps.H"
#include "ImageSet.H"
#include "Pyramid_Ops.H"
#include "Timer.H"
#include "ModelManager.H"
#include <cstdlib>
#include <time.h>
#include <unistd.h>
#include <signal.h>
```

Include dependency graph for nerd-camTCP2.C:

### Defines

- #define **sml** 2
- #define **delta\_min** 3
- #define **delta\_max** 4
- #define **level\_min** 0
- #define **level\_max** 2
- #define **maxdepth** (level\_max + delta\_max + 1)
- #define **normtyp** (VCXNORM\_MAXNORM)
- #define **NBOUT** 10
- #define **IWEIGHT** 1.0
- #define **CWEIGHT** 1.0
- #define **OWEIGHT** 1.0
- #define **FWEIGHT** 1.5

## Functions

- void `terminate` (int s)  
*Signal handler (e.g., for control-C).*
- void `computeCMAP` (TCPmessage &msg, const PyramidType ptyp, const float ori, const float coeff, const int slave, SharedPtr< Beowulf > &b)  
*Compute a conspicuity map from an image received in a message.*
- void `computeCMAP2` (TCPmessage &msg, const PyramidType ptyp, const float ori, const float coeff, const int slave, SharedPtr< Beowulf > &b)  
*Compute a conspicuity map from two images received in a message.*
- void `computeCMAP` (const Image< float > &fima, const PyramidType ptyp, const float ori, const float coeff, const int slave, SharedPtr< Beowulf > &b, const int32 id)  
*Compute a conspicuity map from an image.*
- int `main` (const int argc, const char \*\*argv)

## Variables

- bool `goforever` = true  
*Will turn false on interrupt signal.*

### 11.276.1 Detailed Description

A parallel vision slave to use w/ pvisionTCP2master

Definition in file `nerd-camTCP2.C`.

### 11.276.2 Function Documentation

- 11.276.2.1** void `computeCMAP` (const Image< float > &fima, const PyramidType ptyp, const float ori, const float coeff, const int slave, SharedPtr< Beowulf > &b, const int32 id)

Compute a conspicuity map from an image.

Definition at line 342 of file `nerd-camTCP2.C`.

References Image< T >::addBGnoise(), TCPmessage::addImage(), buildPyramidGeneric(), centerSurround(), downSize(), Image< T >::getHeight(), Image< T >::getWidth(), maxNormalize(), MAXNORMMAX, MAXNORMMIN, VCXNORM\_MAXNORM, and ZEROS.

**11.276.2.2 void computeCMAP (TCPmessage & msg, const PyramidType ptyp, const float ori, const float coeff, const int slave, SharedPtr< Beowulf > & b)**

Compute a conspicuity map from an image received in a message.

Definition at line 316 of file nerd-camTCP2.C.

References ASSERT, computeCMAP(), TCPmessage::getElement(), and TCPmessage::getID().

**11.276.2.3 void computeCMAP2 (TCPmessage & msg, const PyramidType ptyp, const float ori, const float coeff, const int slave, SharedPtr< Beowulf > & b)**

Compute a conspicuity map from two images received in a message.

Definition at line 329 of file nerd-camTCP2.C.

References ASSERT, computeCMAP(), TCPmessage::getElement(), and TCPmessage::getID().

**11.276.2.4 void terminate (int s)**

Signal handler (e.g., for control-C).

Definition at line 62 of file nerd-camTCP2.C.

References goforever.

## 11.276.3 Variable Documentation

**11.276.3.1 bool goforever = true [static]**

Will turn false on interrupt signal.

See the pvisionTCP2go script in bin/ for how to launch the slaves, and see pvisionTCP2-master.C for the master program.

Definition at line 59 of file nerd-camTCP2.C.

Referenced by terminate().

## 11.277 NewClass.C File Reference

```
#include "NewClass.H"
```

Include dependency graph for NewClass.C:

### Variables

- [NewClass](#) `__pad0__`
- [NewClass](#) `const int y`

### 11.277.1 Detailed Description

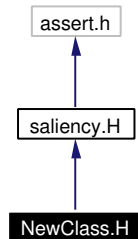
File template for developing a new class

Definition in file [NewClass.C](#).

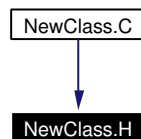
## 11.278 NewClass.H File Reference

```
#include "saliency.H"
```

Include dependency graph for NewClass.H:



This graph shows which files directly or indirectly include this file:



### Classes

- class [NewClass](#)

*Definition and access functions for a new class.*

#### 11.278.1 Detailed Description

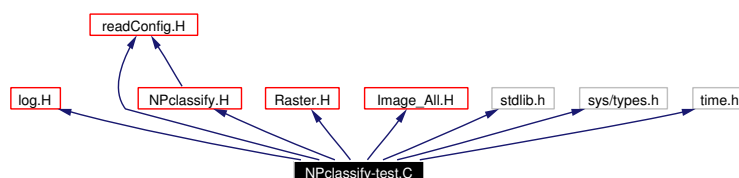
File template for developing a new class

Definition in file [NewClass.H](#).

## 11.279 NPclassify-test.C File Reference

```
#include "log.H"
#include "readConfig.H"
#include "NPclassify.H"
#include "Raster.H"
#include "Image_All.H"
#include <stdlib.h>
#include <sys/types.h>
#include <time.h>
```

Include dependency graph for NPclassify-test.C:



### Functions

- int **main** (int argc, char \*argv[ ])

### Variables

- char \* **configFile**  
*This is the configFile name.*
- **readConfig** **configIn** (25)  
*This is the configFile object.*

### 11.279.1 Detailed Description

Test the non-parametric classifier

Definition in file [NPclassify-test.C](#).

## 11.279.2 Variable Documentation

### 11.279.2.1 `char* configFile`

This is the configFile name.

Definition at line 58 of file NPclassify-test.C.

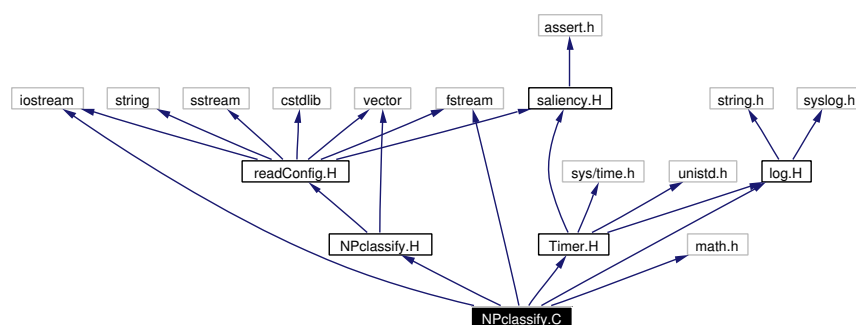
### 11.279.2.2 `readConfig configIn(25)`

This is the configFile object.

## 11.280 NPclassify.C File Reference

```
#include "NPclassify.H"  
#include "Timer.H"  
#include "log.H"  
#include <iostream>  
#include <fstream>  
#include <math.h>
```

Include dependency graph for NPclassify.C:



### 11.280.1 Detailed Description

Test the nonparametric classifier

Definition in file [NPclassify.C](#).

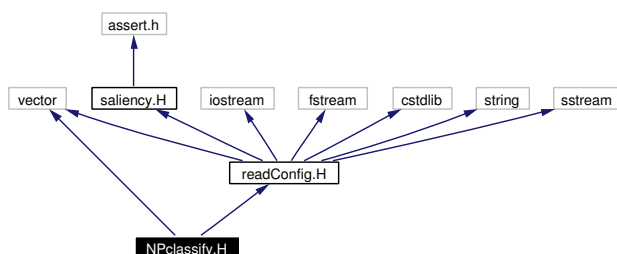


## 11.281 NPclassify.H File Reference

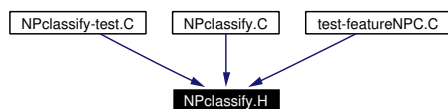
```
#include <vector>
```

```
#include "readConfig.H"
```

Include dependency graph for NPclassify.H:



This graph shows which files directly or indirectly include this file:



### Classes

- class **NPclassify**

#### 11.281.1 Detailed Description

Test the nonparametric classifier

Definition in file [NPclassify.H](#).

## 11.282 NPclassify2.C File Reference

```
#include "NPclassify2.H"
```

```
#include "Timer.H"
```

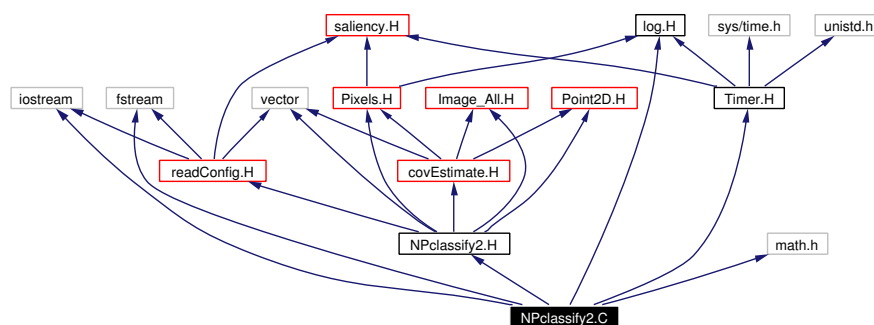
```
#include "log.H"
```

```
#include <iostream>
```

```
#include <fstream>
```

```
#include <math.h>
```

Include dependency graph for NPclassify2.C:



### 11.282.1 Detailed Description

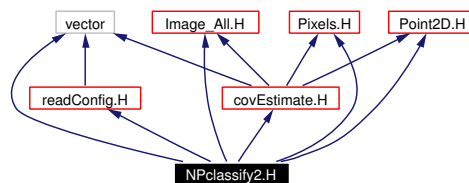
Test the nonparametric classifier

Definition in file [NPclassify2.C](#).

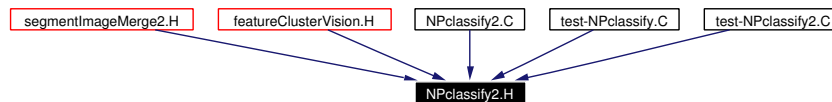
## 11.283 NPclassify2.H File Reference

```
#include <vector>
#include "readConfig.H"
#include "Image_All.H"
#include "Pixels.H"
#include "Point2D.H"
#include "covEstimate.H"
```

Include dependency graph for NPclassify2.H:



This graph shows which files directly or indirectly include this file:



### Classes

- class **NPclassify2**

### 11.283.1 Detailed Description

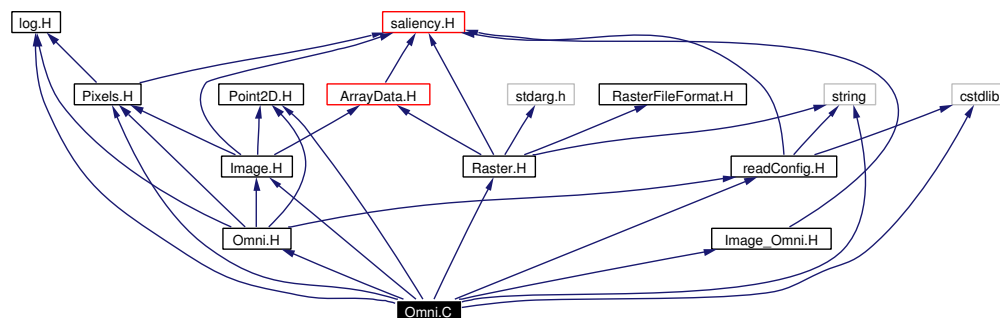
Test the nonparametric classifier

Definition in file [NPclassify2.H](#).

## 11.284 Omni.C File Reference

```
#include "Omni.H"
#include "Image.H"
#include "Image_Omni.H"
#include "log.H"
#include "Pixels.H"
#include "Point2D.H"
#include "Raster.H"
#include "readConfig.H"
#include <stdlib.h>
#include <string.h>
```

Include dependency graph for Omni.C:



## Functions

- int `main` (int argc, char \*argv[ ])

## Variables

- char \* `configFile`  
*This is the configFile name.*
- `readConfig` `configIn` (25)  
*This is the configFile object.*

- `Image< PixRGB< byte > > image`  
*This is an [Image](#) object for RGB images.*
- `Image< PixRGB< byte > > newImage`  
*This is an [Image](#) object for RGB images.*
- `Omni< PixRGB< byte > > omni`  
*This is an [Image](#) object for RGB images.*

### 11.284.1 Detailed Description

Omnidirectional lens distortion correction tool

Definition in file [Omni.C](#).

### 11.284.2 Function Documentation

#### 11.284.2.1 `int main (int argc, char * argv[ ])`

This is the main function for omni directional correction. Its main purpose is to call the omni-directional methods in class [Omni](#). Call from `src2`. You must supply `Omni.conf` as a config file. The command line argument is `"../bin/Omni ../image_in/imagefile"`. Images centers and radius can be calculated by [Omni](#), or specified in the `Omni.conf` file.

Definition at line 79 of file `Omni.C`.

References `configFile`, `configIn`, `image`, `main()`, `newImage`, `omni`, `readConfig::openFile()`, `Raster::ReadRGB()`, `Image< PixRGB< byte > >::resize()`, and `Raster::Visu()`.

### 11.284.3 Variable Documentation

#### 11.284.3.1 `char* configFile`

This is the `configFile` name.

Definition at line 60 of file `Omni.C`.

#### 11.284.3.2 `readConfig configIn(25)`

This is the `configFile` object.

**11.284.3.3   Image< PixRGB<byte> > image**

This is an [Image](#) object for RGB images.

Definition at line 64 of file Omni.C.

Referenced by XWinManaged::drawImage(), main(), and PsychoDisplay::make-BlittableSurface().

**11.284.3.4   Image< PixRGB<byte> > newImage**

This is an [Image](#) object for RGB images.

Definition at line 66 of file Omni.C.

Referenced by main().

**11.284.3.5   Omni< PixRGB<byte> > omni**

This is an [Image](#) object for RGB images.

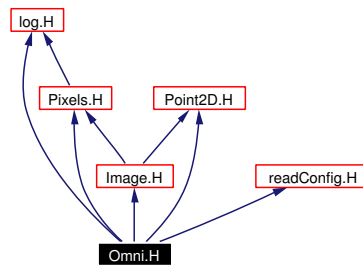
Definition at line 68 of file Omni.C.

Referenced by main().

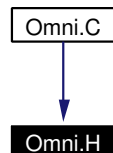
## 11.285 Omni.H File Reference

```
#include "log.H"  
#include "Point2D.H"  
#include "Pixels.H"  
#include "Image.H"  
#include "readConfig.H"
```

Include dependency graph for Omni.H:



This graph shows which files directly or indirectly include this file:



### Classes

- class [Omni](#)  
*omni directional correction tool*

### 11.285.1 Detailed Description

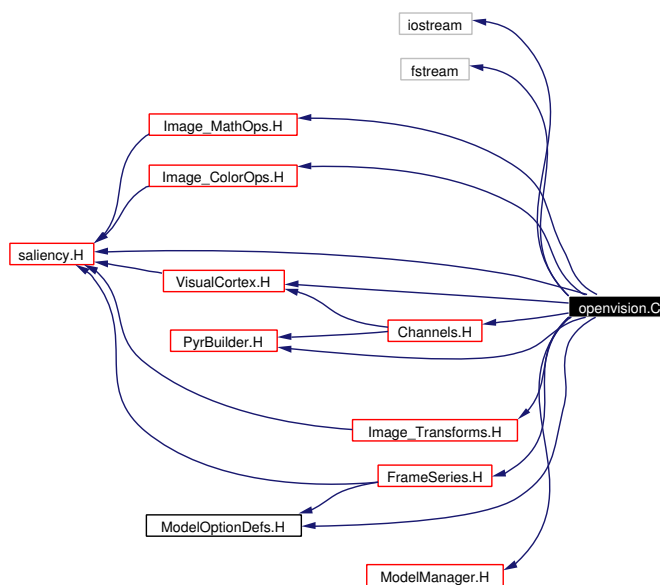
Omni-directional lens distortion correction tool

Definition in file [Omni.H](#).

## 11.286 openvision.C File Reference

```
#include <iostream>
#include <fstream>
#include "saliency.H"
#include "ModelManager.H"
#include "Channels.H"
#include "VisualCortex.H"
#include "FrameSeries.H"
#include "ModelOptionDefs.H"
#include "Image_Transforms.H"
#include "Image_MathOps.H"
#include "Image_ColorOps.H"
#include "PyrBuilder.H"
```

Include dependency graph for openvision.C:



## Defines

- #define NB\_FILTERS 3



- `#define NB_COEFFS 8`

## Functions

- `int main` (`const int argc`, `const char **argv`)

### 11.286.1 Detailed Description

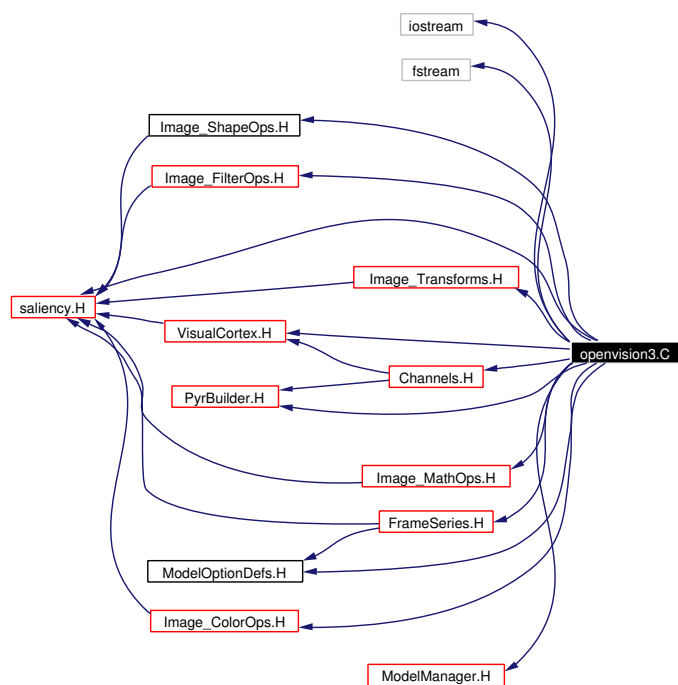
version of [ezvision.C](#) that uses on-file color filters

Definition in file [openvision.C](#).

## 11.287 openvision3.C File Reference

```
#include <iostream>
#include <fstream>
#include "saliency.H"
#include "ModelManager.H"
#include "Channels.H"
#include "VisualCortex.H"
#include "FrameSeries.H"
#include "ModelOptionDefs.H"
#include "Image_Transforms.H"
#include "Image_MathOps.H"
#include "Image_ColorOps.H"
#include "Image_ShapeOps.H"
#include "Image_FilterOps.H"
#include "PyrBuilder.H"
```

Include dependency graph for openvision3.C:



## Defines

- `#define NB_FILTERS 4`
- `#define NB_COEFFS 8`
- `#define RAD 64`
- `#define NB_CONV 1`

## Functions

- `int main (const int argc, const char **argv)`

### 11.287.1 Detailed Description

version of [ezvision.C](#) that uses on-file gray filters

Definition in file [openvision3.C](#).

## 11.288 OptionParser.C File Reference

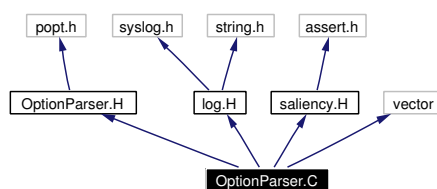
```
#include "OptionParser.H"
```

```
#include "log.H"
```

```
#include "saliency.H"
```

```
#include <vector>
```

Include dependency graph for OptionParser.C:



### Classes

- struct **OptionSet::Impl**

### 11.288.1 Detailed Description

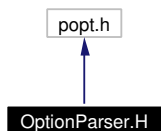
Wrapper of the popt library for parsing command-line

Definition in file [OptionParser.C](#).

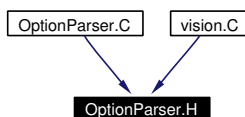
## 11.289 OptionParser.H File Reference

```
#include <popt.h>
```

Include dependency graph for OptionParser.H:



This graph shows which files directly or indirectly include this file:



### Classes

- class [OptionSet](#)  
*Represents a set of available command-line options.*
- class [OptionParser](#)  
*Manages the parsing of argc/argv command-line arguments.*

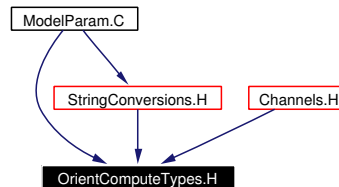
#### 11.289.1 Detailed Description

Wrapper of the popt library for parsing command-line options.

Definition in file [OptionParser.H](#).

## 11.290 OrientComputeTypes.H File Reference

This graph shows which files directly or indirectly include this file:



### Defines

- `#define` [NBORIENTCOMPUTETYPES](#) 3  
*number of known OrientComputeTypes:*

### Enumerations

- `enum` [OrientComputeType](#) { [ORISteerable](#) = 0, [ORIGabor](#) = 1, [ORIGabor-EnergyNorm](#) = 2 }
- Possible types computing orientations.*

### Functions

- `const char *` [orientComputeTypeName](#) (`const` [OrientComputeType](#) p)

#### 11.290.1 Detailed Description

Definition of the pyramid types

Definition in file [OrientComputeTypes.H](#).

#### 11.290.2 Define Documentation

##### 11.290.2.1 `#define` NBORIENTCOMPUTETYPES 3

number of known OrientComputeTypes:

Definition at line 50 of file OrientComputeTypes.H.

Referenced by convertFromString().

### 11.290.3 Enumeration Type Documentation

#### 11.290.3.1 enum [OrientComputeType](#)

Possible types computing orientations.

**Enumeration values:**

***ORISteerable*** use steerable filters

***ORIGabor*** use a Gabor filter

***ORIGaborEnergyNorm*** energy-normalize the image, then use a Gabor filter

Definition at line 42 of file OrientComputeTypes.H.

Referenced by convertFromString(), GaborChannel::GaborChannel(), and GaborChannel::readFrom().

## 11.291 ParamMap.C File Reference

```
#include "ParamMap.H"
#include "log.H"
#include "saliency.H"
#include <fstream>
#include <iomanip>
#include <iostream>
#include <limits>
#include <map>
#include <sstream>
#include <string>
#include <vector>
```

Include dependency graph for ParamMap.C:

### Classes

- struct **Param**
- struct **ParamMap::Impl**
- struct **ParamMap::key\_iterator::IterRep**

### Functions

- void **escapeWrite** (std::ostream &ostrm, const [std::string](#) &s)
- [std::string](#) **escapeRead** (std::istream &istrm)

#### 11.291.1 Detailed Description

I/O operations for parameter files

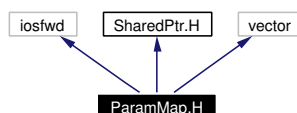
Definition in file [ParamMap.C](#).



## 11.292 ParamMap.H File Reference

```
#include <iosfwd>
#include "SharedPtr.H"
#include <vector>
```

Include dependency graph for ParamMap.H:



This graph shows which files directly or indirectly include this file:

### Classes

- class [ParamMap](#)  
*Allows reading and writing of parameters to and from files or streams.*
- class [ParamMap::Error](#)  
*Exception class thrown by [ParamMap](#) functions in case of error.*
- class [ParamMap::key\\_iterator](#)

### 11.292.1 Detailed Description

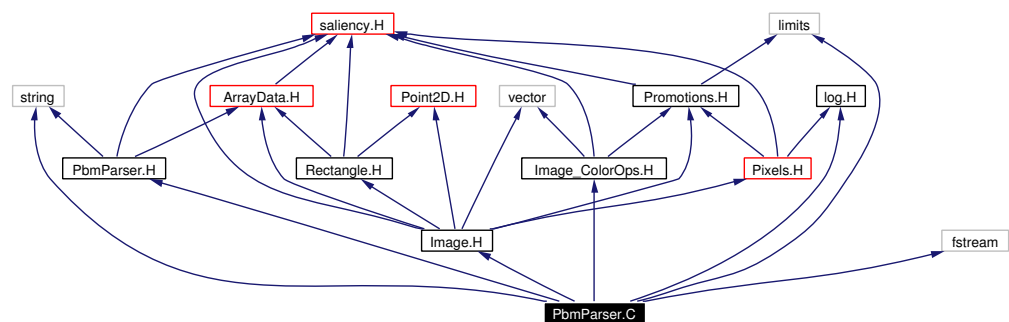
I/O operations for parameter files

Definition in file [ParamMap.H](#).

## 11.293 PbmParser.C File Reference

```
#include "PbmParser.H"
#include "Image.H"
#include "Image_ColorOps.H"
#include "log.H"
#include <fstream>
#include <limits>
#include <string>
```

Include dependency graph for PbmParser.C:



### Classes

- struct **PbmParser::Rep**

### 11.293.1 Detailed Description

Parse pbm/pgm/ppm image files.

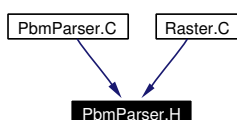
Definition in file [PbmParser.C](#).

## 11.294 PbmParser.H File Reference

```
#include "saliency.H"
#include "ArrayData.H"
#include <string>
```

Include dependency graph for PbmParser.H:

This graph shows which files directly or indirectly include this file:



### Classes

- class [PbmParser](#)

*This class handles reading/parsing of PGM/PPM image files.*

### 11.294.1 Detailed Description

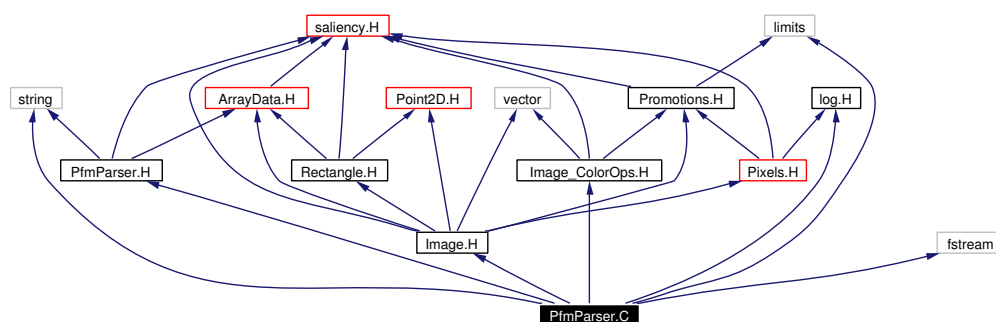
Parse pbm/pgm/ppm image files.

Definition in file [PbmParser.H](#).

## 11.295 PfmParser.C File Reference

```
#include "PfmParser.H"
#include "Image.H"
#include "Image_ColorOps.H"
#include "log.H"
#include <fstream>
#include <limits>
#include <string>
```

Include dependency graph for PfmParser.C:



### Classes

- struct **PfmParser::Rep**

### 11.295.1 Detailed Description

Parse pfm image files.

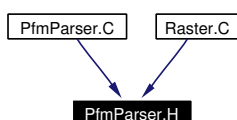
Definition in file [PfmParser.C](#).

## 11.296 PfmParser.H File Reference

```
#include "saliency.H"  
#include "ArrayData.H"  
#include <string>
```

Include dependency graph for PfmParser.H:

This graph shows which files directly or indirectly include this file:



### Classes

- class [PfmParser](#)  
*This class handles reading/parsing of PFM image files.*

### 11.296.1 Detailed Description

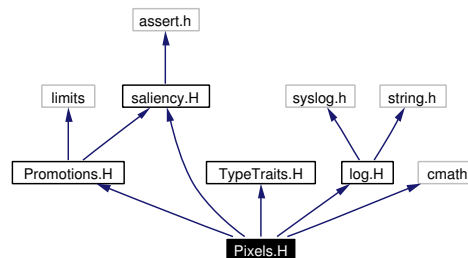
Parse pfm image files.

Definition in file [PfmParser.H](#).

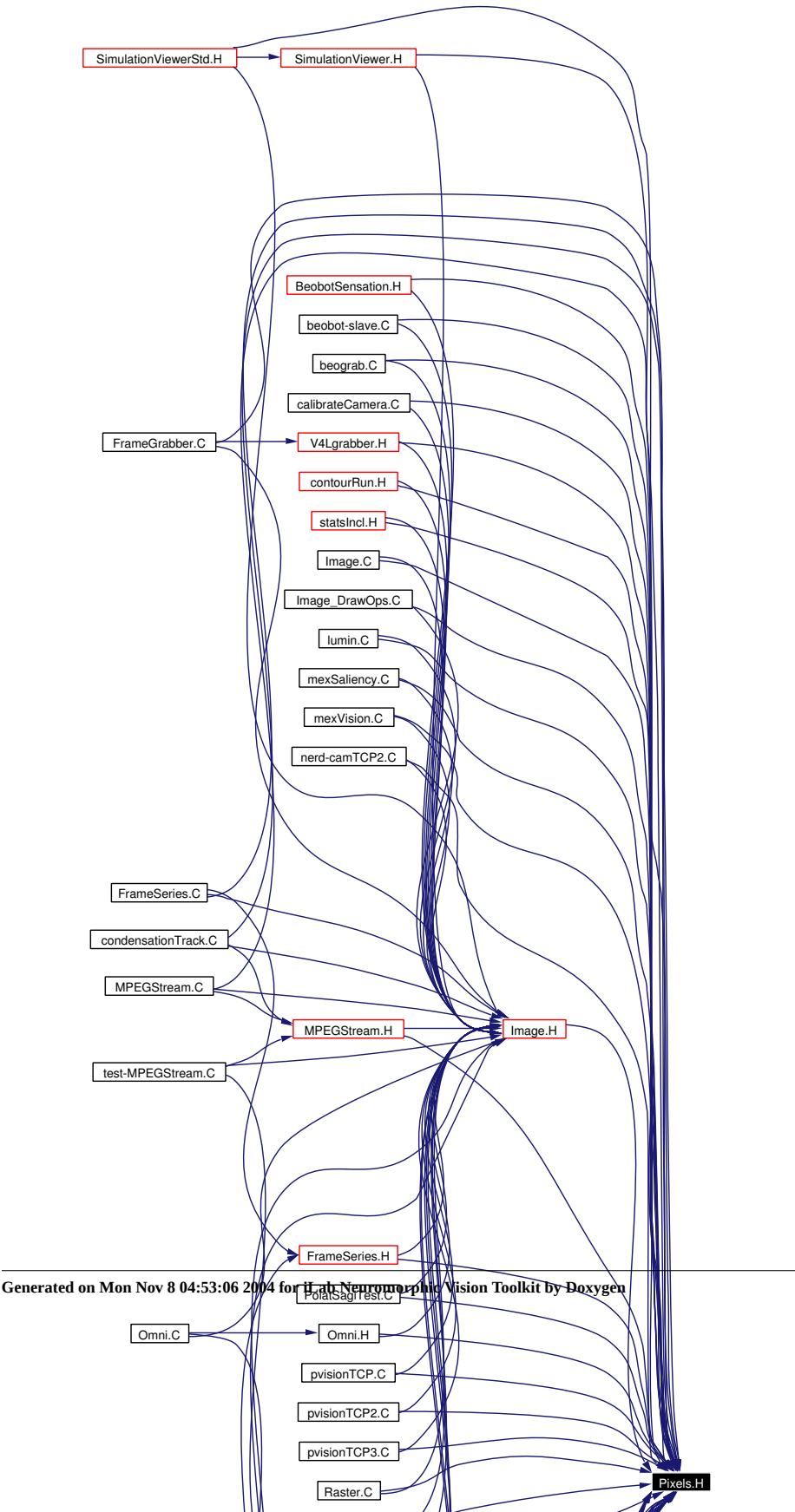
## 11.297 Pixels.H File Reference

```
#include "saliency.H"  
#include "TypeTraits.H"  
#include "log.H"  
#include "Promotions.H"  
#include <cmath>
```

Include dependency graph for Pixels.H:



This graph shows which files directly or indirectly include this file:



## Classes

- struct [TypeTraits< PixRGB< T > >](#)  
Specialize [TypeTraits](#) to indicate that [PixRGB](#) is a "trivial" type.
- class [PixRGB](#)  
Generic pixel class for color RGB pixels.

## Functions

- template<class T> [PixRGB< T > sqrt](#) (const [PixRGB< T > &x](#))  
take square root of r, g, and b:
- template<class T> bool [isFinite](#) (const [PixRGB< T > &x](#))  
Check that all three of r, g, and b have finite values.

### 11.297.1 Detailed Description

Basic pixel types

Definition in file [Pixels.H](#).

### 11.297.2 Function Documentation

#### 11.297.2.1 **template<class T> bool isFinite (const [PixRGB< T > &x](#))** [inline]

Check that all three of r, g, and b have finite values.

Definition at line 637 of file [Pixels.H](#).

References [PixRGB< T >::blue\(\)](#), [PixRGB< T >::green\(\)](#), [isFinite\(\)](#), and [PixRGB< T >::red\(\)](#).

#### 11.297.2.2 **template<class T> [PixRGB< T > sqrt](#) (const [PixRGB< T > &x](#))** [inline]

take square root of r, g, and b:

Definition at line 629 of file [Pixels.H](#).

References [PixRGB< T >::blue\(\)](#), [PixRGB< T >::green\(\)](#), and [PixRGB< T >::red\(\)](#).

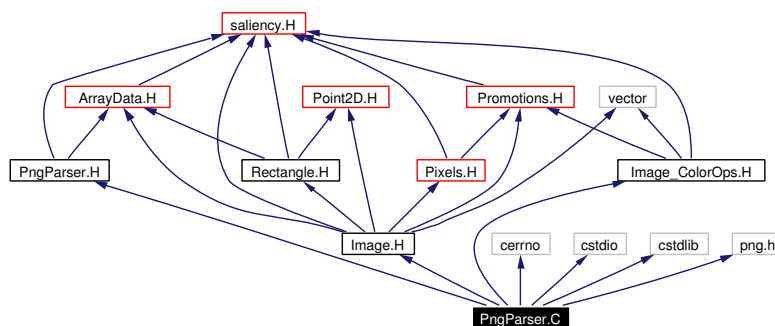


Referenced by avgOrient(), BestEntry::BestEntry(), GAPopulation::compute\_sigma(), MonkeySaccadeController::computeWhenNewDecision(), convolveHmax(), corrpach(), Point2D::distance(), HyperRectangle::distance(), distance(), dog-Filter(), PN03contrastChannel::doInput(), doSqrt(), Image< T >::drawCircle(), Image< T >::drawDisk(), Image< T >::drawDiskCheckTarget(), energyNorm(), ContourNeuronCreate< TC2 >::FindDistance(), fixationMask(), gabor2(), gabor-Filter(), gaborFilter2(), gaborFilter3(), gaussian(), gaussian2D(), ImageSpring< T >::getClusteredImage(), ImageSpring< T >::getDistanceMasses(), Vector3::Get-Normalized(), ImageSpring< T >::getStats(), ImageSpring< T >::getStatsDist(), Image< T >::getStdev(), ScaleSpace::initScaleSpace(), Vector2D::length(), long-RangeExcFilter(), main(), makeImageArray(), maxNormalizeFancy(), maxNormalize-FancyFast(), maxNormalizeFancyLandmark(), maxNormalizeFancyWeak(), KDTree::nearestNeighbor(), KDTree::nearestNeighborList(), Vector3::Norm(), Vector3::Normalize(), FeatureVector::normalize(), omniCorrectGen(), omni-CorrectSp(), Hmax::origGetC2(), VisualCortex::reco(), RMSerr(), ModelEye-HeadController::saccade(), MonkeyEyeHeadController::saccade(), BestEntry::set-DistanceSq(), sqrt(), and SaliencyMap::w().

## 11.298 PngParser.C File Reference

```
#include "PngParser.H"
#include "Image.H"
#include "Image_ColorOps.H"
#include <cerrno>
#include <cstdio>
#include <cstdlib>
#include <png.h>
```

Include dependency graph for PngParser.C:



### Classes

- struct **PngParser::Rep**

### 11.298.1 Detailed Description

Parse png image files.

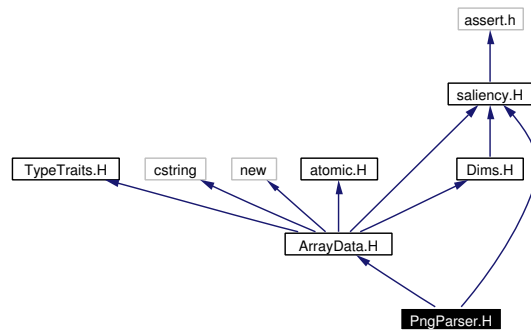
Definition in file [PngParser.C](#).

## 11.299 PngParser.H File Reference

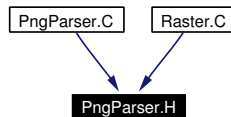
```
#include "saliency.H"
```

```
#include "ArrayData.H"
```

Include dependency graph for PngParser.H:



This graph shows which files directly or indirectly include this file:



### Classes

- class [PngParser](#)  
*This class handles reading/parsing of PNG image files.*

### 11.299.1 Detailed Description

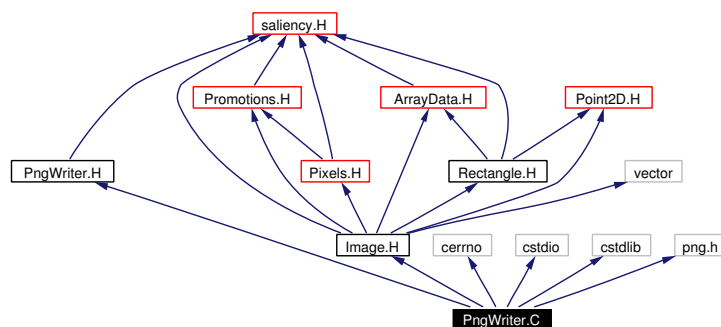
Parse png image files.

Definition in file [PngParser.H](#).

## 11.300 PngWriter.C File Reference

```
#include "PngWriter.H"
#include "Image.H"
#include <cerrno>
#include <cstdio>
#include <cstdlib>
#include <png.h>
```

Include dependency graph for PngWriter.C:



### Classes

- class [PngWriter](#)  
*Helper class for PNG-writing functions.*

### 11.300.1 Detailed Description

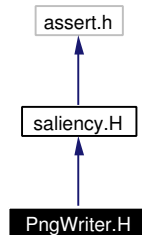
Write PNG image files

Definition in file [PngWriter.C](#).

## 11.301 PngWriter.H File Reference

```
#include "saliency.H"
```

Include dependency graph for PngWriter.H:



This graph shows which files directly or indirectly include this file:



### Namespaces

- namespace [Png](#)

#### 11.301.1 Detailed Description

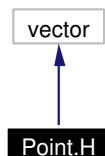
Write PNG image files

Definition in file [PngWriter.H](#).

## 11.302 Point.H File Reference

```
#include <vector>
```

Include dependency graph for Point.H:



### Classes

- class [Point](#)

### Functions

- template<class T, int N> bool **operator==** (const [Point](#)< T, N > &ca1, const [Point](#)< T, N > &ca2)
- template<class T, int N> bool **operator!=** (const [Point](#)< T, N > &p1, const [Point](#)< T, N > &p2)

### 11.302.1 Detailed Description

class for Euclidean vectors, affine points etc

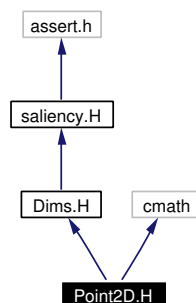
Definition in file [Point.H](#).

## 11.303 Point2D.H File Reference

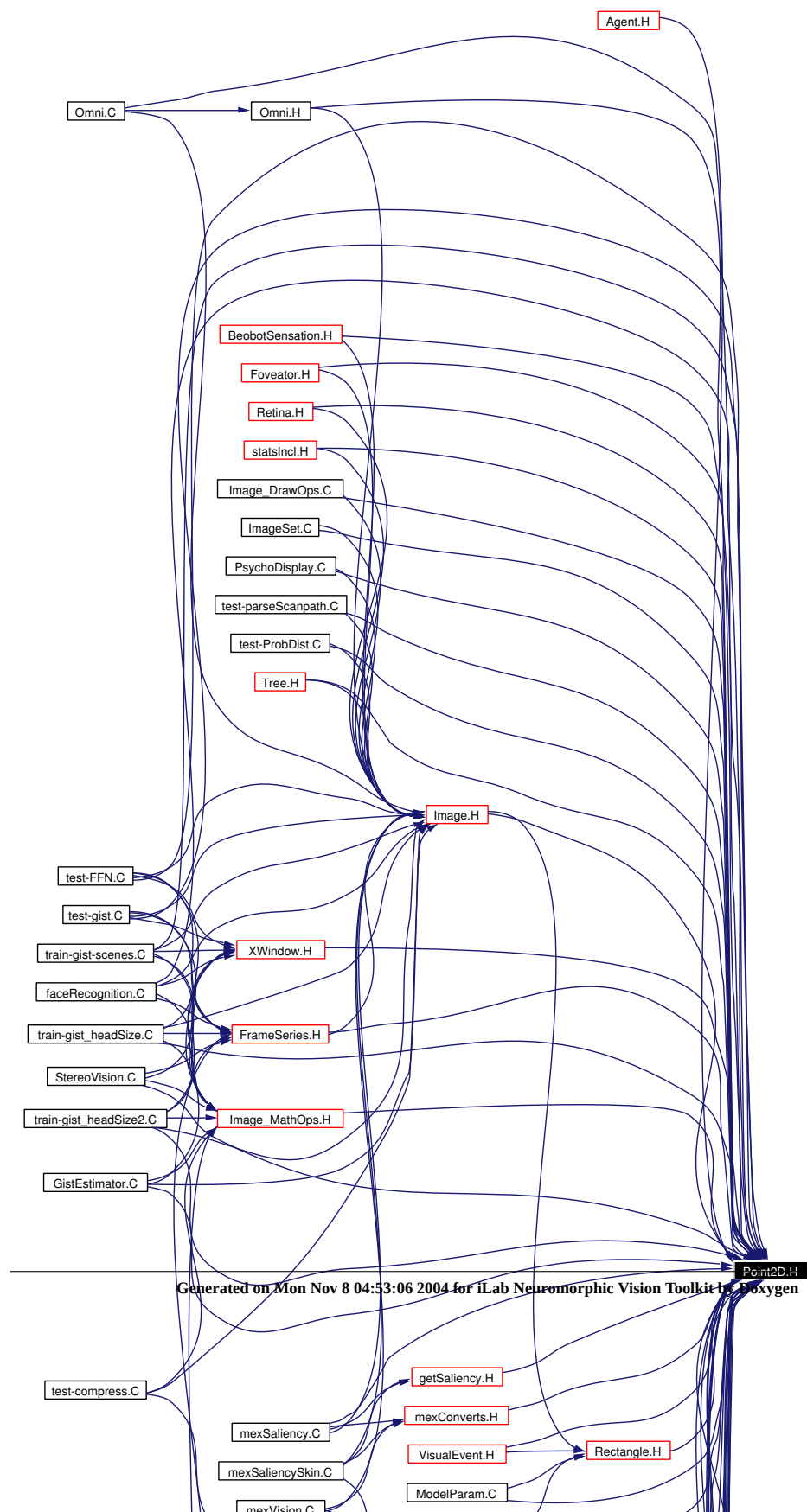
```
#include "Dims.H"
```

```
#include <cmath>
```

Include dependency graph for Point2D.H:



This graph shows which files directly or indirectly include this file:





## Classes

- class [Point2D](#)

*This is a basic class to encode 2D integer coordinates.*

## Functions

- bool `operator==` (const [Point2D](#) &p1, const [Point2D](#) &p2)  
*== operator*
- bool `operator!=` (const [Point2D](#) &p1, const [Point2D](#) &p2)  
*!= operator*

### 11.303.1 Detailed Description

A basic 2D point class

Definition in file [Point2D.H](#).

### 11.303.2 Function Documentation

#### 11.303.2.1 `bool operator!=` (const [Point2D](#) & p1, const [Point2D](#) & p2) [inline]

*!= operator*

Definition at line 202 of file [Point2D.H](#).

References [Point2D::i](#), and [Point2D::j](#).

#### 11.303.2.2 `bool operator==` (const [Point2D](#) & p1, const [Point2D](#) & p2) [inline]

*== operator*

Definition at line 198 of file [Point2D.H](#).

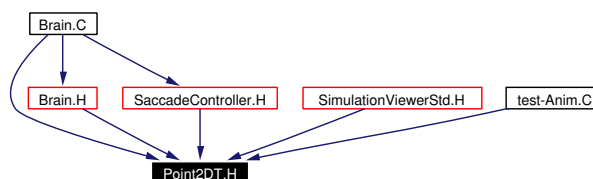
References [Point2D::i](#), and [Point2D::j](#).

## 11.304 Point2DT.H File Reference

```
#include "Point2D.H"
```

Include dependency graph for Point2DT.H:

This graph shows which files directly or indirectly include this file:



### Classes

- class [Point2DT](#)

*This is a fairly trivial (i, j, t) set representing a timestamped [Point2D](#).*

### 11.304.1 Detailed Description

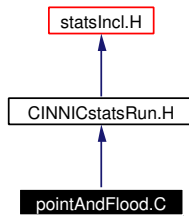
a retinotopic visual fixation

Definition in file [Point2DT.H](#).

## 11.305 pointAndFlood.C File Reference

```
#include "CINNICstatsRun.H"
```

Include dependency graph for pointAndFlood.C:



### Functions

- int [main](#) (int argc, char \*argv[ ])   
*This is a package to run tests on the output from [CINNIC](#).*

### Variables

- char \* [configFile](#)   
*this contains the name of the config file*
- char \* [rawImage](#)   
*this file contains the input RAW image*
- char \* [salImage](#)   
*this file contains a salincy map*
- [readConfig](#) [ReadConfig](#) (25)   
*This is the configFile object.*
- CINNICstatsRun [runStats](#)   
*main object for stats*

### 11.305.1 Detailed Description

find N most salient points in some image for [CINNIC](#)

Definition in file [pointAndFlood.C](#).

## 11.305.2 Function Documentation

### 11.305.2.1 `int main (int argc, char * argv [ ])`

This is a package to run tests on the output from [CINNIC](#).

Definition at line 65 of file [pointAndFlood.C](#).

References [configFile](#), [luminance\(\)](#), [main\(\)](#), [readConfig::openFile\(\)](#), [rawImage](#), [ReadConfig](#), [Raster::ReadGray\(\)](#), [Raster::ReadRGB\(\)](#), [runStats](#), and [salImage](#).

## 11.305.3 Variable Documentation

### 11.305.3.1 `char* configFile`

this contains the name of the config file

Definition at line 50 of file [pointAndFlood.C](#).

### 11.305.3.2 `char* rawImage`

this file contains the input RAW image

Definition at line 53 of file [pointAndFlood.C](#).

Referenced by [main\(\)](#).

### 11.305.3.3 `readConfig ReadConfig(25)`

This is the [configFile](#) object.

Referenced by [main\(\)](#).

### 11.305.3.4 `CINNICstatsRun runStats`

main object for stats

Definition at line 62 of file [pointAndFlood.C](#).

### 11.305.3.5 `char* salImage`

this file contains a salincy map

Definition at line 56 of file pointAndFlood.C.

Referenced by main().

## 11.306 PolatSagiTest.C File Reference

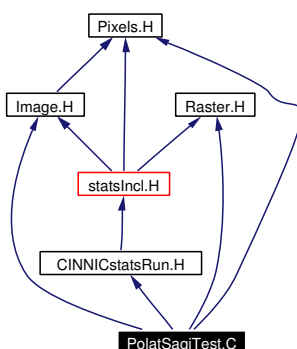
```
#include "CINNICstatsRun.H"
```

```
#include "Raster.H"
```

```
#include "Image.H"
```

```
#include "Pixels.H"
```

Include dependency graph for PolatSagiTest.C:



### Functions

- int [main](#) (int argc, char \*argv[ ])

*This is a package to run tests on the output from [CINNIC](#).*

### Variables

- char \* [configFile](#)  
*this contains the name of the config file*
- char \* [configFile2](#)  
*this contains the name of the second config file*
- char \* [targetFileName](#)  
*this is the target file to open*
- char \* [notargetFileName](#)  
*this is the non target file to open*

- [readConfig ReadConfig](#) (25)  
*This is the configFile object.*
- [readConfig ReadConfig2](#) (25)  
*This is the second configFile object.*
- CINNICstatsRun [runStats](#)  
*main object for stats*
- [Image](#)< float > [Fimage](#) [2]  
*image float*
- float [forReturn](#)  
*float for return*
- [string](#) [in](#)  
*input string for parsing input files*
- int [height](#)  
*image input data*
- int [width](#)  
*image input data*
- int [currentX](#)  
*image input data*
- int [currentY](#)  
*image input data*

### 11.306.1 Detailed Description

FIXME

Definition in file [PolatSagiTest.C](#).

### 11.306.2 Function Documentation

#### 11.306.2.1 int main (int argc, char \* argv[ ])

This is a package to run tests on the output from [CINNIC](#).

Definition at line 33 of file PolatSagiTest.C.

References ASSERT, configFile, configFile2, currentX, currentY, Fimage, for-Return, main(), notargetFileName, readConfig::openFile(), ReadConfig, ReadConfig2, Image< float >::resize(), runStats, Image< float >::setVal(), and targetFileName.

### 11.306.3 Variable Documentation

#### 11.306.3.1 char\* **configFile**

this contains the name of the config file

Definition at line 9 of file PolatSagiTest.C.

#### 11.306.3.2 char\* **configFile2**

this contains the name of the second config file

Definition at line 11 of file PolatSagiTest.C.

#### 11.306.3.3 int **currentX**

image input data

Definition at line 31 of file PolatSagiTest.C.

Referenced by main().

#### 11.306.3.4 int **currentY**

image input data

Definition at line 31 of file PolatSagiTest.C.

Referenced by main().

#### 11.306.3.5 **Image<float> Fimage[2]**

image float

Definition at line 23 of file PolatSagiTest.C.

Referenced by main().



**11.306.3.6 float [forReturn](#)**

float for return

Definition at line 25 of file PolatSagiTest.C.

Referenced by main().

**11.306.3.7 int [height](#)**

image input data

Definition at line 29 of file PolatSagiTest.C.

Referenced by Foveator::Foveator().

**11.306.3.8 string [in](#)**

input string for parsing input files

Definition at line 27 of file PolatSagiTest.C.

**11.306.3.9 char\* [notargetFileName](#)**

this is the non target file to open

Definition at line 15 of file PolatSagiTest.C.

Referenced by main().

**11.306.3.10 [readConfig](#) [ReadConfig](#)(25)**

This is the configFile object.

**11.306.3.11 [readConfig](#) [ReadConfig2](#)(25)**

This is the second configFile object.

Referenced by main().

**11.306.3.12 CINNICstatsRun [runStats](#)**

main object for stats

Definition at line 21 of file PolatSagiTest.C.

**11.306.3.13** char\* **targetFileName**

this is the target file to open

Definition at line 13 of file PolatSagiTest.C.

Referenced by main().

**11.306.3.14** int **width**

image input data

Definition at line 29 of file PolatSagiTest.C.

Referenced by Foveator::Foveator().

## 11.307 prepare\_target\_image.C File Reference

```
#include "Image.H"  
#include "Image_ColorOps.H"  
#include "log.H"  
#include "Raster.H"
```

Include dependency graph for prepare\_target\_image.C:



### Functions

- int **main** (int argc, char \*\*argv)

#### 11.307.1 Detailed Description

Convert 24bit image to 8-bit, col 255 unused

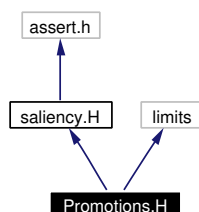
Definition in file [prepare\\_target\\_image.C](#).

## 11.308 Promotions.H File Reference

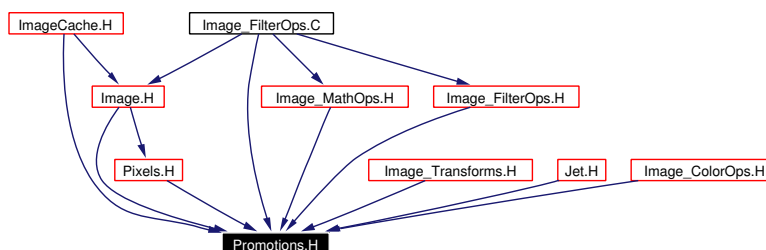
```
#include "saliency.H"
```

```
#include <limits>
```

Include dependency graph for Promotions.H:



This graph shows which files directly or indirectly include this file:



## Classes

- struct [promote\\_trait](#)

*Promote from T1 and T2 to a type than can hold  $T1 * T2$ .*

- struct **promote\_trait**< byte, byte >
- struct **promote\_trait**< byte, int16 >
- struct **promote\_trait**< byte, int32 >
- struct **promote\_trait**< byte, float >
- struct **promote\_trait**< byte, double >
- struct **promote\_trait**< int16, byte >
- struct **promote\_trait**< int16, int16 >
- struct **promote\_trait**< int16, int32 >
- struct **promote\_trait**< int16, float >

- struct **promote\_trait**< int16, double >
- struct **promote\_trait**< int32, float >
- struct **promote\_trait**< int32, double >
- struct **promote\_trait**< float, double >
- struct **promote\_trait**< PixRGB< byte >, byte >
- struct **promote\_trait**< PixRGB< byte >, int16 >
- struct **promote\_trait**< PixRGB< byte >, int32 >
- struct **promote\_trait**< PixRGB< byte >, float >
- struct **promote\_trait**< PixRGB< byte >, double >
- struct **promote\_trait**< PixRGB< int16 >, byte >
- struct **promote\_trait**< PixRGB< int16 >, int16 >
- struct **promote\_trait**< PixRGB< int16 >, int32 >
- struct **promote\_trait**< PixRGB< int16 >, float >
- struct **promote\_trait**< PixRGB< int16 >, double >
- struct **promote\_trait**< PixRGB< int32 >, float >
- struct **promote\_trait**< PixRGB< int32 >, double >
- struct **promote\_trait**< PixRGB< float >, double >
- struct **promoteFromTo**
- struct **promoteFromTo**< T, T >
- struct **promoteFromTo**< int16, byte >
- struct **promoteFromTo**< int32, byte >
- struct **promoteFromTo**< float, byte >
- struct **promoteFromTo**< double, byte >
- struct **promoteFromTo**< int32, int16 >
- struct **promoteFromTo**< float, int16 >
- struct **promoteFromTo**< double, int16 >
- struct **promoteFromTo**< float, int32 >
- struct **promoteFromTo**< double, int32 >
- struct **promoteFromTo**< double, float >
- struct **convert\_helper**< T, TT, false, true >
- struct **convert\_helper**< T, TT, false, false >
- struct **convert\_helper**< T, TT, true, true >
- struct **convert\_helper**< T, TT, true, false >

## Functions

- template<class T, class TT> T **clamped\_convert** (TT x)

### 11.308.1 Detailed Description

basic definitions for template type promotions

Definition in file [Promotions.H](#).

## 11.309 pSNR.C File Reference

```
#include "Image_All.H"
```

```
#include "Raster.H"
```

```
#include "log.H"
```

```
#include <iostream>
```

Include dependency graph for pSNR.C:

### Functions

- int **main** (int argc, char \*\*argv)

### 11.309.1 Detailed Description

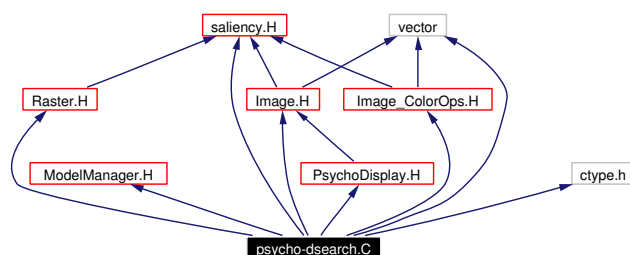
Compute peak signal-to-noise ratio between two images

Definition in file [pSNR.C](#).

## 11.310 psycho-dsearch.C File Reference

```
#include "saliency.H"
#include "ModelManager.H"
#include "PsychoDisplay.H"
#include "Image.H"
#include "Raster.H"
#include "Image_ColorOps.H"
#include <ctype.h>
#include <vector>
```

Include dependency graph for psycho-dsearch.C:



### Defines

- `#define NMASK 10`  
*number of frames in the mask*

### Functions

- `int main (const int argc, const char **argv)`

#### 11.310.1 Detailed Description

Psychophysics display for a search for two targets that are presented to the observer prior to the search

Definition in file [psycho-dsearch.C](#).

## 11.310.2 Define Documentation

### 11.310.2.1 `#define NMASK 10`

number of frames in the mask

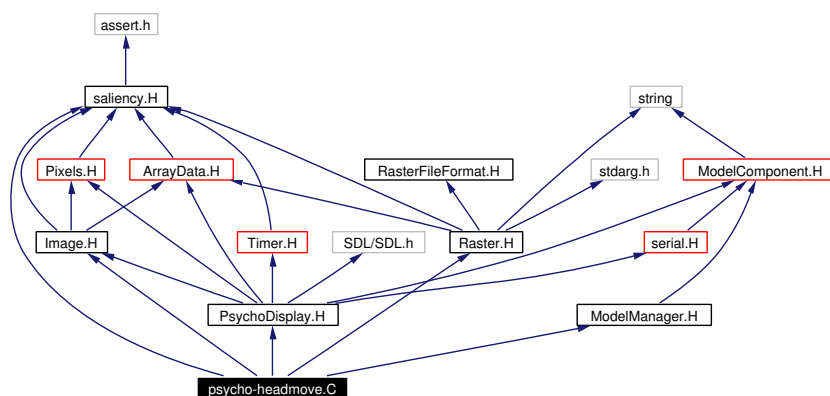
Definition at line 48 of file psycho-dsearch.C.



## 11.311 psycho-headmove.C File Reference

```
#include "saliency.H"
#include "ModelManager.H"
#include "PsychoDisplay.H"
#include "Image.H"
#include "Raster.H"
```

Include dependency graph for psycho-headmove.C:



### Functions

- int **main** (const int argc, const char \*\*argv)

#### 11.311.1 Detailed Description

Psychophysics display of head movement stimuli

Definition in file [psycho-headmove.C](#).

## 11.312 psycho-movie.C File Reference

```
#include "saliency.H"
#include "ModelManager.H"
#include "PsychoDisplay.H"
#include "Image.H"
#include "MPEGStream.H"
#include <SDL/SDL.h>
#include <deque>
```

Include dependency graph for psycho-movie.C:

### Defines

- `#define CACHELEN 150`

### Functions

- `bool cacheFrame` ([SharedPtr](#)< [MPEGStream](#) > &mp, [std::deque](#)< `byte` \* > &cache)
- `int main` (const int argc, const char \*\*argv)

### 11.312.1 Detailed Description

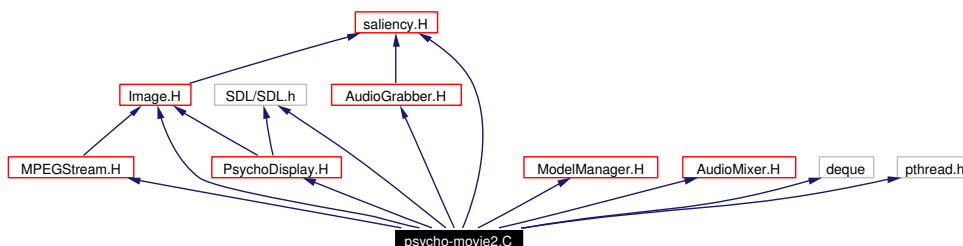
Psychophysics display of movies

Definition in file [psycho-movie.C](#).

## 11.313 psycho-movie2.C File Reference

```
#include "saliency.H"
#include "ModelManager.H"
#include "PsychoDisplay.H"
#include "Image.H"
#include "MPEGStream.H"
#include "AudioGrabber.H"
#include "AudioMixer.H"
#include <SDL/SDL.h>
#include <deque>
#include <pthread.h>
```

Include dependency graph for psycho-movie2.C:



## Defines

- #define **CACHELEN** 150
- #define **BLACKPAUSE** true

## Functions

- void \* **audiorecorder** (void \*agbv)
- bool **cacheFrame** ([SharedPtr](#)< [MPEGStream](#) > &mp, [std::deque](#)< [byte](#) \* > &cache)
- int **main** (const int argc, const char \*\*argv)

## Variables

- volatile bool **recordaudio** = false
- volatile bool **keepgoing** = true

### 11.313.1 Detailed Description

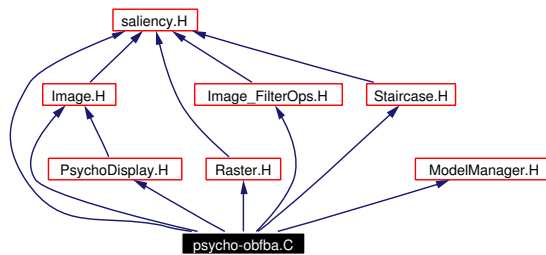
Psychophysics interactive display of movies

Definition in file [psycho-movie2.C](#).

## 11.314 psycho-obfba.C File Reference

```
#include "saliency.H"
#include "ModelManager.H"
#include "PsychoDisplay.H"
#include "Image.H"
#include "Raster.H"
#include "Image_FilterOps.H"
#include "Staircase.H"
```

Include dependency graph for psycho-obfba.C:



### Classes

- struct [PsychoTask](#)  
*A summary struct to handle task settings.*

### Enumerations

- enum [TaskType](#) { **None** = 0, **Orientation** = 1, **Drift** = 2, **Blank** = 3 }  
*Types of tasks.*

### Functions

- void [parseTaskDefinition](#) (const [std::string](#) &arg, [PsychoTask](#) &p)  
*Parse task definitions given at command line.*
- int **main** (const int argc, const char \*\*argv)

## Variables

- const bool `blocking` = false

*Set this to true if you want to block until responses are given.*

- bool `shadow` = true

*Set this to true if you want to add drop shadows.*

### 11.314.1 Detailed Description

Psychophysics display

Definition in file `psycho-obfba.C`.

### 11.314.2 Enumeration Type Documentation

#### 11.314.2.1 enum `TaskType`

Types of tasks.

Definition at line 52 of file `psycho-obfba.C`.

### 11.314.3 Function Documentation

#### 11.314.3.1 void `parseTaskDefinition` (const `std::string` & *arg*, `PsychoTask` & *p*)

Parse task definitions given at command line.

Definition at line 63 of file `psycho-obfba.C`.

References `PsychoTask::drift`, `parseTaskDefinition()`, `PsychoTask::s`, `PsychoTask::theta`, and `PsychoTask::tt`.

Referenced by `parseTaskDefinition()`.

### 11.314.4 Variable Documentation

#### 11.314.4.1 const bool `blocking` = false

Set this to true if you want to block until responses are given.

Definition at line 46 of file `psycho-obfba.C`.

**11.314.4.2 bool `shadow` = true**

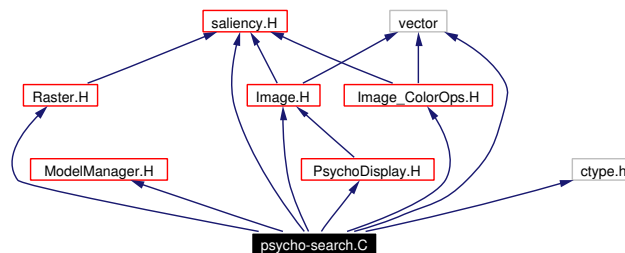
Set this to true if you want to add drop shadows.

Definition at line 49 of file psycho-obfba.C.

## 11.315 psycho-search.C File Reference

```
#include "saliency.H"
#include "ModelManager.H"
#include "PsychoDisplay.H"
#include "Image.H"
#include "Raster.H"
#include "Image_ColorOps.H"
#include <ctype.h>
#include <vector>
```

Include dependency graph for psycho-search.C:



### Defines

- `#define NMASK 10`  
*number of frames in the mask*

### Functions

- `int main (const int argc, const char **argv)`

#### 11.315.1 Detailed Description

Psychophysics display for a search for a target that is presented to the observer prior to the search

Definition in file [psycho-search.C](#).



## 11.315.2 Define Documentation

### 11.315.2.1 `#define NMASK 10`

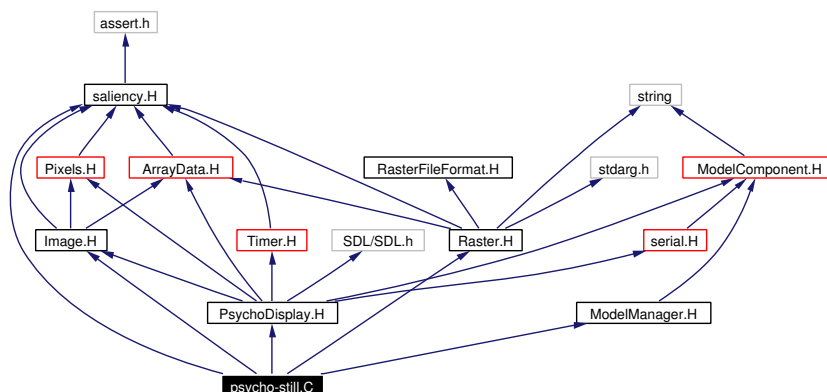
number of frames in the mask

Definition at line 48 of file psycho-search.C.

## 11.316 psycho-still.C File Reference

```
#include "saliency.H"
#include "ModelManager.H"
#include "PsychoDisplay.H"
#include "Image.H"
#include "Raster.H"
```

Include dependency graph for psycho-still.C:



### Functions

- int **main** (const int argc, const char \*\*argv)

#### 11.316.1 Detailed Description

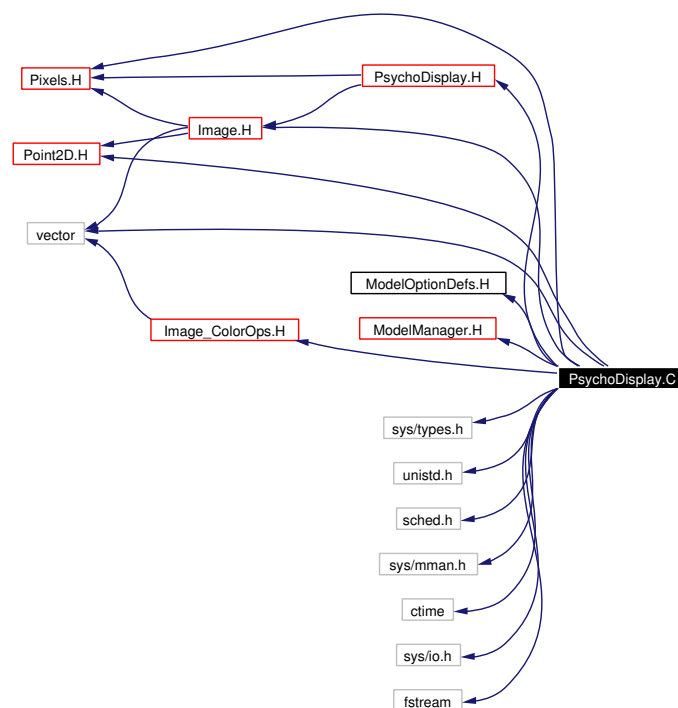
Psychophysics display of still images and check response to prevent cheating

Definition in file [psycho-still.C](#).

## 11.317 PsychoDisplay.C File Reference

```
#include "PsychoDisplay.H"
#include "ModelOptionDefs.H"
#include "ModelManager.H"
#include "Point2D.H"
#include "Pixels.H"
#include "Image.H"
#include "Image_ColorOps.H"
#include <sys/types.h>
#include <unistd.h>
#include <sched.h>
#include <sys/mman.h>
#include <ctime>
#include <sys/io.h>
#include <fstream>
#include <vector>
```

Include dependency graph for PsychoDisplay.C:



### 11.317.1 Detailed Description

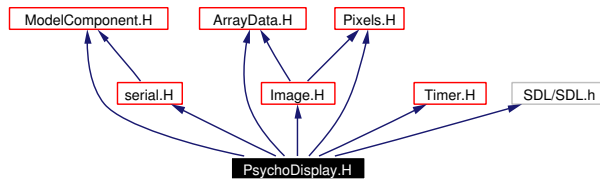
Display psychophysics stimuli

Definition in file [PsychoDisplay.C](#).

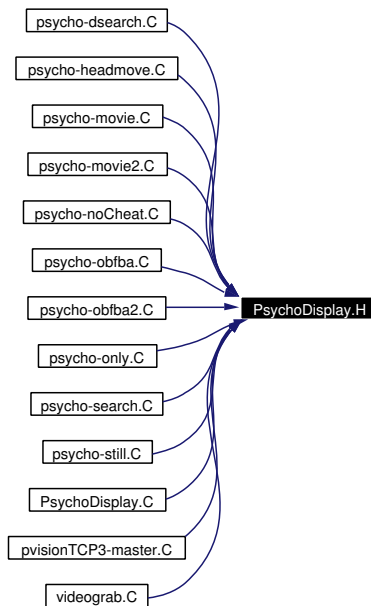
## 11.318 PsychoDisplay.H File Reference

```
#include "ModelComponent.H"  
#include "ArrayData.H"  
#include "Timer.H"  
#include "serial.H"  
#include "Image.H"  
#include "Pixels.H"  
#include <SDL/SDL.h>
```

Include dependency graph for PsychoDisplay.H:



This graph shows which files directly or indirectly include this file:



## Classes

- class [PsychoDisplay](#)  
*Class fo do various psychophysics displays.*
- struct **PsychoDisplay::PsychoEvent**

### 11.318.1 Detailed Description

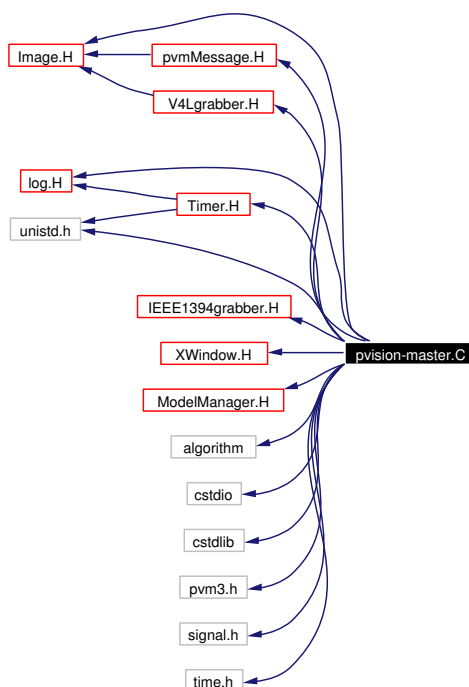
Display psychophysics stimuli

Definition in file [PsychoDisplay.H](#).

## 11.319 pvision-master.C File Reference

```
#include "Image.H"
#include "log.H"
#include "pvmMessage.H"
#include "Timer.H"
#include "V4Lgrabber.H"
#include "IEEE1394grabber.H"
#include "XWindow.H"
#include "ModelManager.H"
#include <algorithm>
#include <cstdio>
#include <cstdlib>
#include <pvm3.h>
#include <signal.h>
#include <time.h>
#include <unistd.h>
```

Include dependency graph for pvision-master.C:



## Defines

- `#define NAVG 20`

## Functions

- void **terminate** (int s)
- int **main** (const int argc, const char \*\*argv)

## Variables

- bool **goforever** = true

### 11.319.1 Detailed Description

Grab video frames & process over beowulf with pvision

Definition in file [pvision-master.C](#).



## 11.320 pvision.C File Reference

```
#include "fancynorm.H"
#include "saliency.H"
#include "pvmMessage.H"
#include "Image.H"
#include "log.H"
#include "VisualCortex.H"
#include "Raster.H"
#include "SingleChannel.H"
#include "ModelManager.H"
#include <algorithm>
#include <cstdlib>
#include <iostream>
#include <pvm3.h>
#include <signal.h>
#include <time.h>
#include <unistd.h>
```

Include dependency graph for pvision.C:

### Defines

- #define [WRITE\\_RESULTS\\_OUT](#)
- #define **WRITE\_RESULT\_PATH** "/home/tmp/1"

### Functions

- void **srnd48** (long int)
- void **terminate** (int s)
- int **main** (const int argc, const char \*\*argv)

### Variables

- bool **goforever** = true

### 11.320.1 Detailed Description

A parallel vision processing slave

Definition in file [pvision.C](#).

### 11.320.2 Define Documentation

#### 11.320.2.1 `#define WRITE_RESULTS_OUT`

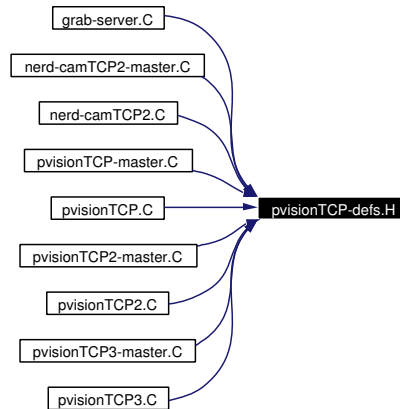
This is a basic round-robin processing scheme: different frames are fully processed by different CPUs.

- wait for image from PVM parent
- process it
- send trajectory image back to master
- start again from the beginning

Definition at line 64 of file [pvision.C](#).

## 11.321 pvisionTCP-defs.H File Reference

This graph shows which files directly or indirectly include this file:



### Defines

- #define **BEO\_RETINA** 1
- #define **BEO\_WINNER** 2
- #define **BEO\_LUMINANCE** 3
- #define **BEO\_REDGREEN** 4
- #define **BEO\_BLUEYELLOW** 5
- #define **BEO\_ORI0** 6
- #define **BEO\_ORI45** 7
- #define **BEO\_ORI90** 8
- #define **BEO\_ORI135** 9
- #define **BEO\_CMAP** 10
- #define **BEO\_FLICKER** 11
- #define **NBCMAP** 7  
*Number of conspicuity maps for pvisionTCP.*
- #define **NBCMAP2** 8  
*Number of conspicuity maps fo pvisionTCP2.*
- #define **POFFSET** 8  
*Node offset for the two processing streams:.*

### 11.321.1 Detailed Description

Definitions for pvisionTCP & pvisionTCPmaster

Definition in file [pvisionTCP-defs.H](#).

### 11.321.2 Define Documentation

#### 11.321.2.1 `#define BEO_RETINA 1`

Various definitions for the parallel saliency code.

Definition at line 41 of file pvisionTCP-defs.H.

Referenced by main().

#### 11.321.2.2 `#define NBCMAP 7`

Number of conspicuity maps for pvisionTCP.

Definition at line 54 of file pvisionTCP-defs.H.

#### 11.321.2.3 `#define NBCMAP2 8`

Number of conspicuity maps for pvisionTCP2.

Definition at line 57 of file pvisionTCP-defs.H.

Referenced by receiveCMAPS().

#### 11.321.2.4 `#define POFFSET 8`

Node offset for the two processing streams:.

Definition at line 60 of file pvisionTCP-defs.H.

## 11.322 pvisionTCP-master.C File Reference

```
#include "pvisionTCP-defs.H"
#include "saliency.H"
#include "Beowulf.H"
#include "Image_All.H"
#include "ImageSet.H"
#include "XWindow.H"
#include "V4Lgrabber.H"
#include "Timer.H"
#include "ModelManager.H"
#include <algorithm>
#include <signal.h>
#include <unistd.h>
```

Include dependency graph for pvisionTCP-master.C:

### Defines

- #define **NAVG** 20  
*Number of frames over which average framerate is computed.*
- #define **NSTAT** 6  
*Number of stats collected.*
- #define **PRESCALE** 2  
*prescale level by which we downsize images before sending them off*

### Functions

- void **terminate** (int s)  
*Signal handler (e.g., for control-C).*
- int **main** (const int argc, const char \*\*argv)

## Variables

- bool `goforever` = true

*Will turn false on interrupt signal.*

### 11.322.1 Detailed Description

Grab video & process over beowulf w/ pvisionTCP

Definition in file [pvisionTCP-master.C](#).

### 11.322.2 Define Documentation

#### 11.322.2.1 `#define NAVG 20`

Number of frames over which average framerate is computed.

Definition at line 57 of file pvisionTCP-master.C.

#### 11.322.2.2 `#define NSTAT 6`

Number of stats collected.

Definition at line 59 of file pvisionTCP-master.C.

#### 11.322.2.3 `#define PRESCALE 2`

prescale level by which we downsize images before sending them off

Definition at line 62 of file pvisionTCP-master.C.

### 11.322.3 Function Documentation

#### 11.322.3.1 `void terminate (int s)`

Signal handler (e.g., for control-C).

Definition at line 67 of file pvisionTCP-master.C.

References `goforever`.

## 11.322.4 Variable Documentation

### 11.322.4.1 bool `goforever` = true [static]

Will turn false on interrupt signal.

Definition at line 64 of file pvisionTCP-master.C.

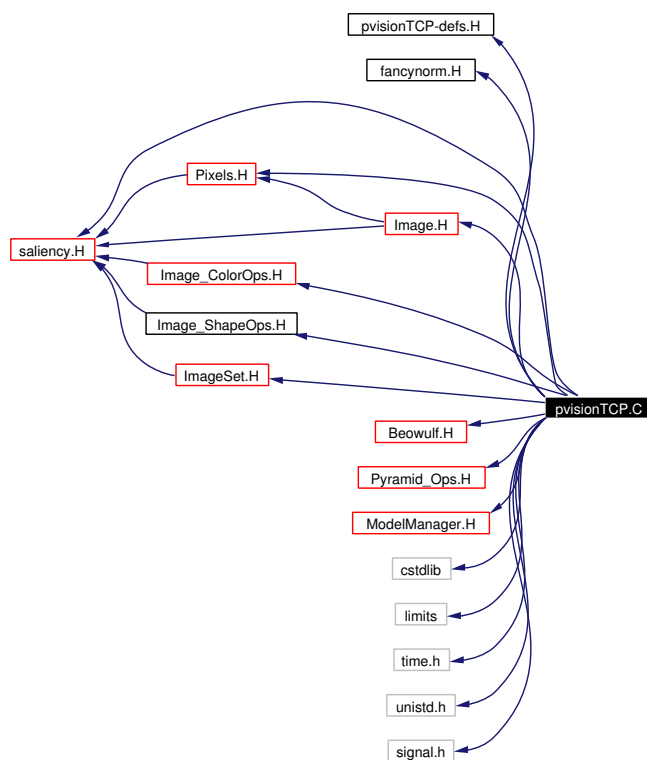
Referenced by `terminate()`.

## 11.323 pvisionTCP.C File Reference

```
#include "pvisionTCP-defs.H"
#include "fancynorm.H"
#include "saliency.H"
#include "Beowulf.H"
#include "Pixels.H"
#include "Image_ColorOps.H"
#include "Image_ShapeOps.H"
#include "ImageSet.H"
#include "Pyramid_Ops.H"
#include "ModelManager.H"
#include "Image.H"
#include <cstdlib>
#include <limits>
#include <time.h>
#include <unistd.h>
#include <signal.h>
```

Include dependency graph for pvisionTCP.C:





## Defines

- #define **sml** 2
- #define **delta\_min** 3
- #define **delta\_max** 4
- #define **level\_min** 0
- #define **level\_max** 2
- #define **maxdepth** (level\_max + delta\_max + 1)
- #define **normtyp** (VCXNORM\_MAXNORM)
- #define **NBOUT** 10
- #define **IWEIGHT** 1.0
- #define **CWEIGHT** 1.0
- #define **OWEIGHT** 1.0
- #define **FWEIGHT** 1.0

## Functions

- void [terminate](#) (int s)

*Signal handler (e.g., for control-C).*

- void `computeCMAP` (`TCPmessage` &msg, const `PyramidType` ptyp, const float ori, const float coeff, const int slave, `SharedPtr`< `Beowulf` > &b)

*Compute a conspicuity map from an image received in a message.*

- void `computeCMAP2` (`TCPmessage` &msg, const `PyramidType` ptyp, const float ori, const float coeff, const int slave, `SharedPtr`< `Beowulf` > &b)

*Compute a conspicuity map from two images received in a message.*

- void `computeCMAP` (const `Image`< float > &fima, const `PyramidType` ptyp, const float ori, const float coeff, const int slave, `SharedPtr`< `Beowulf` > &b, const `int32` id)

*Compute a conspicuity map from an image.*

- int `main` (const int argc, const char \*\*argv)

## Variables

- bool `goforever` = true

*Will turn false on interrupt signal.*

### 11.323.1 Detailed Description

A parallel vision slave to use with `pvisionTCPmaster`

Definition in file `pvisionTCP.C`.

### 11.323.2 Function Documentation

- 11.323.2.1** void `computeCMAP` (const `Image`< float > &fima, const `PyramidType` ptyp, const float ori, const float coeff, const int slave, `SharedPtr`< `Beowulf` > &b, const `int32` id)

Compute a conspicuity map from an image.

- 11.323.2.2** void `computeCMAP` (`TCPmessage` & msg, const `PyramidType` ptyp, const float ori, const float coeff, const int slave, `SharedPtr`< `Beowulf` > &b)

Compute a conspicuity map from an image received in a message.

**11.323.2.3** void computeCMAP2 (TCPmessage & msg, const PyramidType ptyp, const float ori, const float coeff, const int slave, SharedPtr<Beowulf > & b)

Compute a conspicuity map from two images received in a message.

**11.323.2.4** void terminate (int s)

Signal handler (e.g., for control-C).

Definition at line 62 of file pvisionTCP.C.

References goforever.

### 11.323.3 Variable Documentation

**11.323.3.1** bool goforever = true [static]

Will turn false on interrupt signal.

See the pvisionTCPgo script in bin/ for how to launch the slaves, and see pvisionTCP-master.C for the master program.

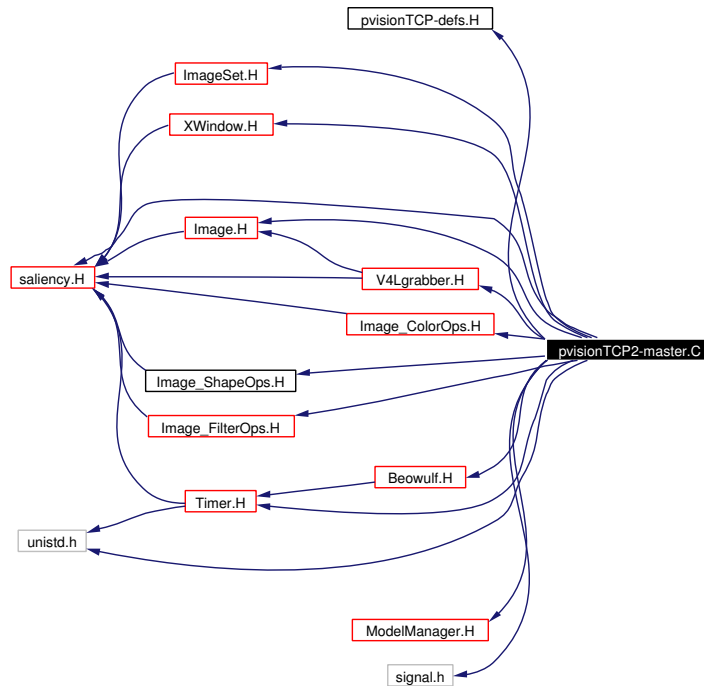
Definition at line 59 of file pvisionTCP.C.

Referenced by terminate().

## 11.324 pvisionTCP2-master.C File Reference

```
#include "pvisionTCP-defs.H"
#include "saliency.H"
#include "Beowulf.H"
#include "Image.H"
#include "Image_ColorOps.H"
#include "Image_ShapeOps.H"
#include "Image_FilterOps.H"
#include "ImageSet.H"
#include "XWindow.H"
#include "V4Lgrabber.H"
#include "Timer.H"
#include "ModelManager.H"
#include <unistd.h>
#include <signal.h>
```

Include dependency graph for pvisionTCP2-master.C:



## Defines

- `#define NAVG 20`  
*Number of frames over which average framerate is computed.*
- `#define NSTAT 4`  
*Number of stats collected.*
- `#define PRESCALE 2`  
*prescale level by which we downsize images before sending them off*
- `#define SHIFTINT 5`  
*shift attention at most once every SHIFTINT frames*
- `#define SMINJECT 0.6f`  
*injection ratio between new and older sm:*
- `#define SMTHRESH 200`  
*threshold salience (between 0 and 255) to warrant a shift of attention*

## Functions

- void `terminate` (int s)  
*Signal handler (e.g., for control-C).*
- int `main` (const int argc, const char \*\*argv)

## Variables

- bool `goforever` = true  
*Will turn false on interrupt signal.*

### 11.324.1 Detailed Description

Grab & process over beowulf w/ pvisionTCP2

Definition in file [pvisionTCP2-master.C](#).

### 11.324.2 Define Documentation

#### 11.324.2.1 `#define NAVG 20`

Number of frames over which average framerate is computed.

This parallel vision processing master is for use with pvisionTCP2. See the pvision-TCP2go script in bin/ for how to launch the slaves.

Definition at line 58 of file pvisionTCP2-master.C.

#### 11.324.2.2 `#define NSTAT 4`

Number of stats collected.

Definition at line 60 of file pvisionTCP2-master.C.

#### 11.324.2.3 `#define PRESCALE 2`

prescale level by which we downsize images before sending them off

Definition at line 63 of file pvisionTCP2-master.C.

**11.324.2.4 #define SHIFTINT 5**

shift attention at most once every SHIFTINT frames

Definition at line 66 of file pvisionTCP2-master.C.

**11.324.2.5 #define SMINJECT 0.6f**

injection ratio between new and older sm:

Definition at line 69 of file pvisionTCP2-master.C.

**11.324.2.6 #define SMTHRESH 200**

threshold salience (between 0 and 255) to warrant a shift of attention

Definition at line 72 of file pvisionTCP2-master.C.

**11.324.3 Function Documentation****11.324.3.1 void terminate (int s)**

Signal handler (e.g., for control-C).

Definition at line 77 of file pvisionTCP2-master.C.

References goforever.

**11.324.4 Variable Documentation****11.324.4.1 bool goforever = true [static]**

Will turn false on interrupt signal.

Definition at line 74 of file pvisionTCP2-master.C.

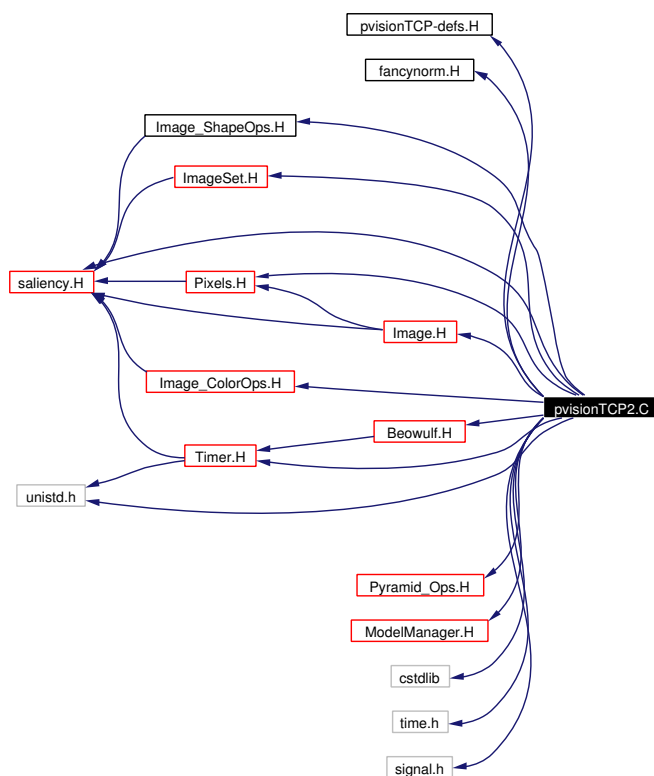
Referenced by terminate().

## 11.325 pvisionTCP2.C File Reference

```
#include "pvisionTCP-defs.H"
#include "fancynorm.H"
#include "saliency.H"
#include "Beowulf.H"
#include "Pixels.H"
#include "Image.H"
#include "Image_ColorOps.H"
#include "Image_ShapeOps.H"
#include "ImageSet.H"
#include "Pyramid_Ops.H"
#include "Timer.H"
#include "ModelManager.H"
#include <cstdlib>
#include <time.h>
#include <unistd.h>
#include <signal.h>
```

Include dependency graph for pvisionTCP2.C:





## Defines

- `#define sml 2`
- `#define delta_min 3`
- `#define delta_max 4`
- `#define level_min 0`
- `#define level_max 2`
- `#define maxdepth (level_max + delta_max + 1)`
- `#define normtyp (VCXNORM_MAXNORM)`
- `#define NBOUT 10`
- `#define IWEIGHT 1.0`
- `#define CWEIGHT 1.0`
- `#define OWEIGHT 1.0`
- `#define FWEIGHT 1.5`

## Functions

- void `terminate` (int s)

*Signal handler (e.g., for control-C).*

- void `computeCMAP` (`TCPmessage` &msg, const `PyramidType` ptyp, const float ori, const float coeff, const int slave, `SharedPtr`< `Beowulf` > &b)

*Compute a conspicuity map from an image received in a message.*

- void `computeCMAP2` (`TCPmessage` &msg, const `PyramidType` ptyp, const float ori, const float coeff, const int slave, `SharedPtr`< `Beowulf` > &b)

*Compute a conspicuity map from two images received in a message.*

- void `computeCMAP` (const `Image`< float > &fima, const `PyramidType` ptyp, const float ori, const float coeff, const int slave, `SharedPtr`< `Beowulf` > &b, const `int32` id)

*Compute a conspicuity map from an image.*

- int `main` (const int argc, const char \*\*argv)

## Variables

- bool `goforever` = true

*Will turn false on interrupt signal.*

### 11.325.1 Detailed Description

A parallel vision slave to use w/ `pvisionTCP2master`

Definition in file `pvisionTCP2.C`.

### 11.325.2 Function Documentation

- 11.325.2.1** void `computeCMAP` (const `Image`< float > &fima, const `PyramidType` ptyp, const float ori, const float coeff, const int slave, `SharedPtr`< `Beowulf` > &b, const `int32` id)

Compute a conspicuity map from an image.

- 11.325.2.2** void `computeCMAP` (`TCPmessage` & msg, const `PyramidType` ptyp, const float ori, const float coeff, const int slave, `SharedPtr`< `Beowulf` > &b)

Compute a conspicuity map from an image received in a message.

**11.325.2.3** `void computeCMAP2 (TCPmessage & msg, const PyramidType ptyp, const float ori, const float coeff, const int slave, SharedPtr< Beowulf > & b)`

Compute a conspicuity map from two images received in a message.

**11.325.2.4** `void terminate (int s)`

Signal handler (e.g., for control-C).

Definition at line 62 of file pvisionTCP2.C.

References `goforever`.

### 11.325.3 Variable Documentation

**11.325.3.1** `bool goforever = true` [static]

Will turn false on interrupt signal.

See the `pvisionTCP2go` script in `bin/` for how to launch the slaves, and see `pvisionTCP2-master.C` for the master program.

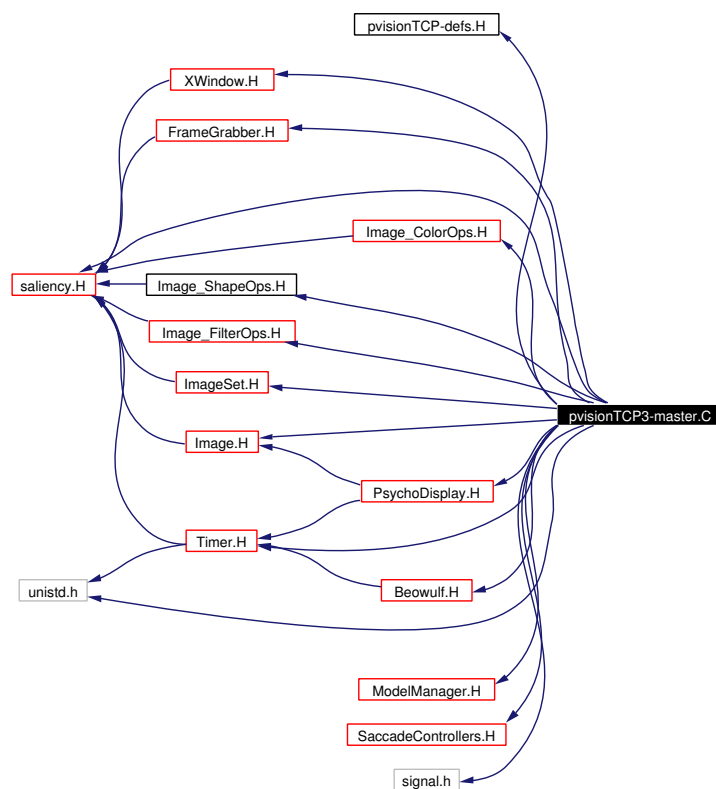
Definition at line 59 of file `pvisionTCP2.C`.

Referenced by `terminate()`.

## 11.326 pvisionTCP3-master.C File Reference

```
#include "pvisionTCP-defs.H"
#include "saliency.H"
#include "Beowulf.H"
#include "Image.H"
#include "Image_ColorOps.H"
#include "Image_ShapeOps.H"
#include "Image_FilterOps.H"
#include "ImageSet.H"
#include "XWindow.H"
#include "FrameGrabber.H"
#include "Timer.H"
#include "ModelManager.H"
#include "PsychoDisplay.H"
#include "SaccadeControllers.H"
#include <unistd.h>
#include <signal.h>
```

Include dependency graph for pvisionTCP3-master.C:



## Defines

- #define `NAVG` 20  
Number of frames over which average framerate is computed.
- #define `PRESCALE` 2  
prescale level by which we downsize images before sending them off

## Functions

- void `terminate` (int s)  
Signal handler (e.g., for control-C).
- void `receiveCMAPS` (SharedPtr< Beowulf > &beo, Image< float > \*cmap, int32 \*cmapframe)  
function to receive results from slaves

- int **main** (const int argc, const char \*\*argv)

## Variables

- bool `goforever` = true  
*Will turn false on interrupt signal.*

### 11.326.1 Detailed Description

Grab & process over beowulf w/ pvisionTCP3

Definition in file [pvisionTCP3-master.C](#).

### 11.326.2 Define Documentation

#### 11.326.2.1 `#define NAVG 20`

Number of frames over which average framerate is computed.

This parallel vision processing master is for use with pvisionTCP3. See the pvisionTCP3go script in bin/ for how to launch the slaves. The main difference between the TCP2 and TCP3 versions is that TCP3 uses load-balancing to send images to the slaves, and the master node is not only in charge of grabbing the video and displaying the results, but also of collecting the various conspicuity maps and assembling them into the saliency map.

Definition at line 64 of file pvisionTCP3-master.C.

#### 11.326.2.2 `#define PRESCALE 2`

prescale level by which we downsize images before sending them off

Definition at line 67 of file pvisionTCP3-master.C.

### 11.326.3 Function Documentation

#### 11.326.3.1 `void receiveCMAPS (SharedPtr< Beowulf > & beo, Image< float > * cmap, int32 * cmapframe)`

function to receive results from slaves

Definition at line 309 of file pvisionTCP3-master.C.

References ASSERT, TCPmessage::getElement(), and NBCMAP2.

#### 11.326.3.2 void terminate (int s)

Signal handler (e.g., for control-C).

Definition at line 72 of file pvisionTCP3-master.C.

References goforever.

### 11.326.4 Variable Documentation

#### 11.326.4.1 bool goforever = true [static]

Will turn false on interrupt signal.

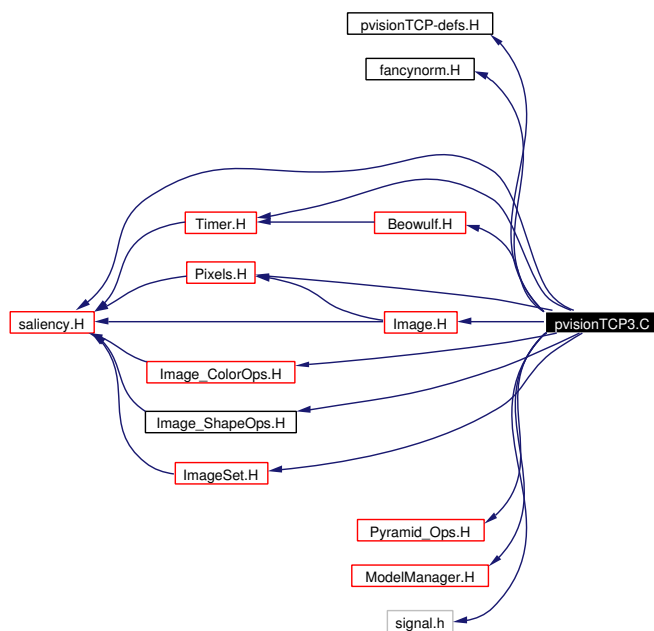
Definition at line 69 of file pvisionTCP3-master.C.

Referenced by terminate().

## 11.327 pvisionTCP3.C File Reference

```
#include "pvisionTCP-defs.H"  
#include "fancynorm.H"  
#include "saliency.H"  
#include "Beowulf.H"  
#include "Pixels.H"  
#include "Image.H"  
#include "Image_ColorOps.H"  
#include "Image_ShapeOps.H"  
#include "ImageSet.H"  
#include "Pyramid_Ops.H"  
#include "Timer.H"  
#include "ModelManager.H"  
#include <signal.h>
```

Include dependency graph for pvisionTCP3.C:





## Defines

- #define **sml** 2
- #define **delta\_min** 3
- #define **delta\_max** 4
- #define **level\_min** 0
- #define **level\_max** 2
- #define **maxdepth** (level\_max + delta\_max + 1)
- #define **normtyp** (VCXNORM\_MAXNORM)
- #define **IWEIGHT** 1.0
- #define **CWEIGHT** 1.0
- #define **OWEIGHT** 1.0
- #define **FWEIGHT** 1.5

## Functions

- void **terminate** (int s)  
*Signal handler (e.g., for control-C).*
- void **computeCMAP** (TCPmessage &msg, const PyramidType ptyp, const float ori, const float coeff, const int slave, SharedPtr< Beowulf > &b)  
*Compute a conspicuity map from an image received in a message.*
- void **computeCMAP2** (TCPmessage &msg, const PyramidType ptyp, const float ori, const float coeff, const int slave, SharedPtr< Beowulf > &b)  
*Compute a conspicuity map from two images received in a message.*
- void **computeCMAP** (const Image< float > &fima, const PyramidType ptyp, const float ori, const float coeff, const int slave, SharedPtr< Beowulf > &b, const int32 id)  
*Compute a conspicuity map from an image.*
- int **main** (const int argc, const char \*\*argv)

## Variables

- bool **goforever** = true  
*Will turn false on interrupt signal.*

### 11.327.1 Detailed Description

A parallel vision slave to use w/ pvisionTCP3master

Definition in file [pvisionTCP3.C](#).

### 11.327.2 Function Documentation

**11.327.2.1** `void computeCMAP (const Image< float > & fima, const PyramidType ptyp, const float ori, const float coeff, const int slave, SharedPtr< Beowulf > & b, const int32 id)`

Compute a conspicuity map from an image.

Referenced by `computeCMAP()`, and `computeCMAP2()`.

**11.327.2.2** `void computeCMAP (TCPmessage & msg, const PyramidType ptyp, const float ori, const float coeff, const int slave, SharedPtr< Beowulf > & b)`

Compute a conspicuity map from an image received in a message.

**11.327.2.3** `void computeCMAP2 (TCPmessage & msg, const PyramidType ptyp, const float ori, const float coeff, const int slave, SharedPtr< Beowulf > & b)`

Compute a conspicuity map from two images received in a message.

**11.327.2.4** `void terminate (int s)`

Signal handler (e.g., for control-C).

Definition at line 57 of file `pvisionTCP3.C`.

References `goforever`.

### 11.327.3 Variable Documentation

**11.327.3.1** `bool goforever = true` `[static]`

Will turn false on interrupt signal.

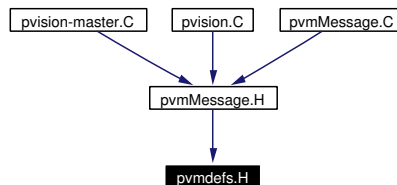
See the `pvisionTCP3go` script in `bin/` for how to launch the slaves, and see `pvisionTCP3-master.C` for the master program.

Definition at line 54 of file pvisionTCP3.C.

Referenced by terminate().

## 11.328 pvmdefs.H File Reference

This graph shows which files directly or indirectly include this file:



### Defines

- `#define` [SYSVSIGNAL](#)  
*as generated by running aimk on our machines...*
- `#define` **NOWAIT3**
- `#define` **RSHCOMMAND** `"/usr/bin/rsh"`
- `#define` **NEEDENDIAN**
- `#define` **FDSETNOTSTRUCT**
- `#define` **HASERRORVARS**
- `#define` **CTIMEISTIMET**
- `#define` **SYSERRISCONST**

### 11.328.1 Detailed Description

Set of PVM definitions

Definition in file [pvmdefs.H](#).

### 11.328.2 Define Documentation

#### 11.328.2.1 `#define` SYSVSIGNAL

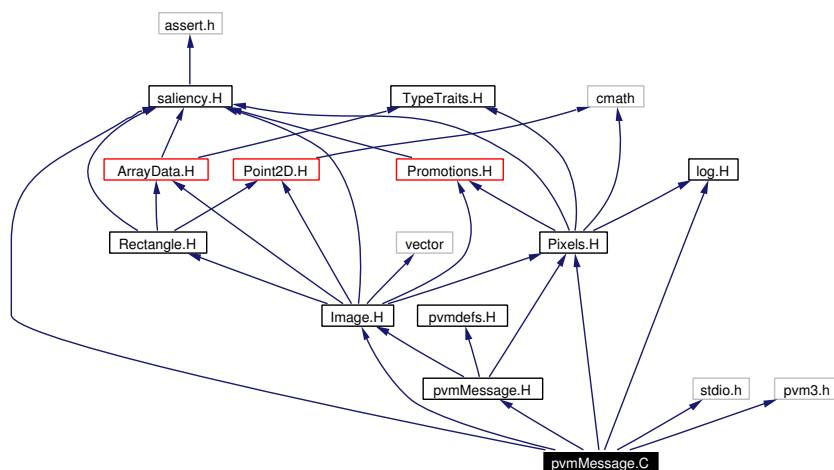
as generated by running aimk on our machines...

Definition at line 40 of file [pvmdefs.H](#).

## 11.329 pvmMessage.C File Reference

```
#include "saliency.H"
#include "pvmMessage.H"
#include <stdio.h>
#include <pvm3.h>
#include "log.H"
#include "Pixels.H"
#include "Image.H"
```

Include dependency graph for pvmMessage.C:



### 11.329.1 Detailed Description

PVM messages to be passed for inter-process communication

Definition in file [pvmMessage.C](#).

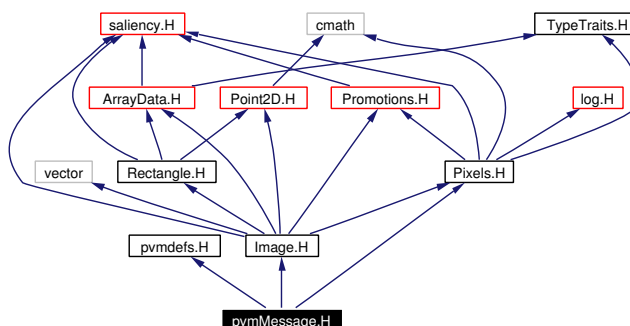
## 11.330 pvmMessage.H File Reference

```
#include "pvmdefs.H"
```

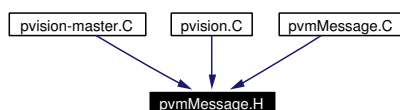
```
#include "Pixels.H"
```

```
#include "Image.H"
```

Include dependency graph for pvmMessage.H:



This graph shows which files directly or indirectly include this file:



## Classes

- class [PvmMessage](#)  
*definition and access functions for PVM messages*

## Defines

- #define [PVMSG\\_END](#) 0  
*types of possible messages:*
- #define [PVMSG\\_COLBYTEIMA](#) 1
- #define [PVMSG\\_DIE](#) 2

- `#define PVMSG_FIXATION 3`
- `#define PVMSG_STRING 4`

### 11.330.1 Detailed Description

Definition and access functions for PVM messages

Definition in file [pvmMessage.H](#).

### 11.330.2 Define Documentation

#### 11.330.2.1 `#define PVMSG_END 0`

types of possible messages:

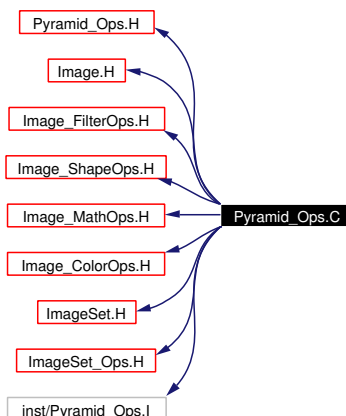
Definition at line 45 of file pvmMessage.H.

Referenced by `PvmMessage::send()`.

### 11.331 Pyramid\_Ops.C File Reference

```
#include "Pyramid_Ops.H"
#include "Image.H"
#include "Image_FilterOps.H"
#include "Image_ShapeOps.H"
#include "Image_MathOps.H"
#include "Image_ColorOps.H"
#include "ImageSet.H"
#include "ImageSet_Ops.H"
#include "inst/Pyramid_Ops.I"
```

Include dependency graph for Pyramid\_Ops.C:



### Functions

- `template<class T> ImageSet< T > buildPyrGaussian (const Image< T > &image, int depth, int filterSize)`
- `template<class T> ImageSet< T > buildPyrConvolve (const Image< T > &image, int depth, const Image< float > &filter)`
- `template<class T> ImageSet< T > buildPyrLaplacian (const Image< T > &image, int depth, int filterSize)`
- `template<class T> ImageSet< T > buildPyrOriented (const Image< T > &image, int depth, int filterSize, float theta, float intens)`
- `template<class T> ImageSet< T > buildPyrLocalAvg (const Image< T > &image, int depth)`



- `template<class T> ImageSet< T > buildPyrLocalMax (const Image< T > &image, int depth)`
- `template<class T> ImageSet< T > buildPyrGeneric (const Image< T > &image, int depth, const PyramidType typ, const float gabor_theta, const float intens)`
- `ImageSet< float > buildPyrGabor (const ImageSet< float > &gaussianPyr, float angle, float filter_period, float elongation, int size, int flags)`  
*Build a Gabor Pyramid with a given angle, filter period and elongation.*
- `ImageSet< float > buildPyrGabor (const Image< float > &img, int depth, float angle, float filter_period, float elongation, int size, int flags)`  
*Build a Gabor Pyramid with a given angle, filter period and elongation.*
- `template<class T> bool isDyadic (const ImageSet< T > &pyr)`  
*Check whether the pyramid is dyadic.*
- `template<class T> Image< T > centerSurround (const ImageSet< T > &pyr, const int lev1, const int lev2, const bool absol, const ImageSet< float > *clipPyr)`  
*Compute center-surround between images at lev1 and lev2.*
- `template<class T> void centerSurround (const ImageSet< T > &pyr, const int lev1, const int lev2, Image< T > &pos, Image< T > &neg, const ImageSet< float > *clipPyr)`  
*Compute center-surround between images, splitting positive and negative.*
- `template<class T> Image< T > centerSurroundSingleOpponent (const ImageSet< T > &cpyr, const ImageSet< T > &spyr, const int lev1, const int lev2, const bool absol, const ImageSet< float > *clipPyr)`  
*Compute single-opponent center-surround between images at lev1 and lev2.*
- `template<class T> void centerSurroundSingleOpponent (const ImageSet< T > &cpyr, const ImageSet< T > &spyr, const int lev1, const int lev2, Image< T > &pos, Image< T > &neg, const ImageSet< float > *clipPyr)`  
*Compute single-opp center-surr between imgs, splitting pos and neg.*
- `template<class T> Image< T > centerSurroundDiff (const ImageSet< T > &pyr1, const ImageSet< T > &pyr2, const int lev1, const int lev2, const bool absol, const ImageSet< float > *clipPyr)`  
*Compute center-surround between difference images at lev1 and lev2.*
- `template<class T> void centerSurroundDiff (const ImageSet< T > &pyr1, const ImageSet< T > &pyr2, const int lev1, const int lev2, Image< T > &pos, Image< T > &neg, const ImageSet< float > *clipPyr)`

*Compute center-surround differences, splitting pos & neg values.*

- `template<class T> Image< T > centerSurroundDiffSingleOpponent (const ImageSet< T > &cpyr1, const ImageSet< T > &cpyr2, const ImageSet< T > &spyr1, const ImageSet< T > &spyr2, const int lev1, const int lev2, const bool absol, const ImageSet< float > *clipPyr)`

*Compute single-opp center-surround btw difference images at lev1 and lev2.*

- `template<class T> void centerSurroundDiffSingleOpponent (const ImageSet< T > &cpyr1, const ImageSet< T > &cpyr2, const ImageSet< T > &spyr1, const ImageSet< T > &spyr2, const int lev1, const int lev2, Image< T > &pos, Image< T > &neg, const ImageSet< float > *clipPyr)`

*Compute single-opp center-surround differences, splitting pos & neg values.*

### 11.331.1 Detailed Description

Free functions operating on pyramid data structures

Definition in file [Pyramid\\_Ops.C](#).

### 11.331.2 Function Documentation

#### 11.331.2.1 `ImageSet<float> buildPyrGabor (const Image< float > &img, int depth, float angle, float filter_period, float elongation = 1.0, int size = -1, int flags = 0)`

Build a Gabor Pyramid with a given angle, filter period and elongation.

##### Parameters:

**size** determines the filter size (in pixels) - in the default case (-1), the size is computed from the filter\_preiod and elongation

Definition at line 295 of file [Pyramid\\_Ops.C](#).

References [buildPyrGaussian\(\)](#), and [buildPyrLaplacian\(\)](#).

Referenced by [GaborPyrBuilder< T >::build\(\)](#), and [GaborPyrBuilder< T >::input\(\)](#).

#### 11.331.2.2 `ImageSet<float> buildPyrGabor (const ImageSet< float > &gaussianPyr, float angle, float filter_period, float elongation = 1.0, int size = -1, int flags = 0)`

Build a Gabor Pyramid with a given angle, filter period and elongation.

**Parameters:**

*size* determines the filter size (in pixels) - in the default case (-1), the size is computed from the filter\_preiod and elongation

Definition at line 225 of file Pyramid\_Ops.C.

**11.331.2.3** `template<class T> void centerSurround (const ImageSet< T > & pyr, const int lev1, const int lev2, Image< T > & pos, Image< T > & neg, const ImageSet< float > * clipPyr)`

Compute center-surround between images, splitting positive and negative.

Definition at line 369 of file Pyramid\_Ops.C.

References ASSERT, ImageSet< T >::getDepth(), ImageSet< T >::isNonEmpty(), and uint.

Referenced by centerSurround(), centerSurroundDiff(), centerSurroundDiffSingleOpponent(), centerSurroundSingleOpponent(), computeCMAP(), and doubleOpp().

**11.331.2.4** `template<class T> Image<T> centerSurround (const ImageSet< T > & pyr, const int lev1, const int lev2, const bool absol, const ImageSet< float > * clipPyr)`

Compute center-surround between images at lev1 and lev2.

Definition at line 342 of file Pyramid\_Ops.C.

**11.331.2.5** `template<class T> void centerSurroundDiff (const ImageSet< T > & pyr1, const ImageSet< T > & pyr2, const int lev1, const int lev2, Image< T > & pos, Image< T > & neg, const ImageSet< float > * clipPyr)`

Compute center-surround differences, splitting pos & neg values.

Definition at line 490 of file Pyramid\_Ops.C.

References ASSERT, centerSurround(), ImageSet< T >::getDepth(), Image< T >::getDims(), ImageSet< T >::isNonEmpty(), and uint.

**11.331.2.6** `template<class T> Image<T> centerSurroundDiff (const ImageSet< T > & pyr1, const ImageSet< T > & pyr2, const int lev1, const int lev2, const bool absol = false, const ImageSet< float > * clipPyr = 0)`

Compute center-surround between difference images at lev1 and lev2.

Difference images are obtained at each level by taking the difference between the image from *\*this* minus the image from *other\_pyr*

Definition at line 456 of file Pyramid\_Ops.C.

**11.331.2.7** `template<class T> void centerSurroundDiffSingleOpponent (const ImageSet< T > & cpyr1, const ImageSet< T > & cpyr2, const ImageSet< T > & spyr1, const ImageSet< T > & spyr2, const int lev1, const int lev2, Image< T > & pos, Image< T > & neg, const ImageSet< float > * clipPyr)`

Compute single-opp center-surround differnces, splitting pos & neg values.

Definition at line 562 of file Pyramid\_Ops.C.

References ASSERT, centerSurround(), ImageSet< T >::getDepth(), Image< T >::getDims(), ImageSet< T >::isNonEmpty(), and uint.

**11.331.2.8** `template<class T> Image<T> centerSurroundDiffSingleOpponent (const ImageSet< T > & cpyr1, const ImageSet< T > & cpyr2, const ImageSet< T > & spyr1, const ImageSet< T > & spyr2, const int lev1, const int lev2, const bool absol = false, const ImageSet< float > * clipPyr = 0)`

Compute single-opp center-surround btw difference images at lev1 and lev2.

Difference images are obtained at each level by taking the difference between the image from *\*this* minus the image from *other\_pyr*

Definition at line 524 of file Pyramid\_Ops.C.

**11.331.2.9** `template<class T> void centerSurroundSingleOpponent (const ImageSet< T > & cpyr, const ImageSet< T > & spyr, const int lev1, const int lev2, Image< T > & pos, Image< T > & neg, const ImageSet< float > * clipPyr)`

Compute single-opp center-surr between imgs, splitting pos and neg.

Definition at line 426 of file Pyramid\_Ops.C.

References ASSERT, centerSurround(), ImageSet< T >::getDepth(), ImageSet< T >::isNonEmpty(), and uint.

**11.331.2.10** `template<class T> Image<T> centerSurroundSingleOpponent  
(const ImageSet< T > & cpyr, const ImageSet< T > & spyr, const  
int lev1, const int lev2, const bool absol = false, const ImageSet<  
float > * clipPyr = 0)`

Compute single-opponent center-surround between images at lev1 and lev2.

This is essentially cpyr[lev1] - spyr[lev2]

Definition at line 395 of file Pyramid\_Ops.C.

**11.331.2.11** `template<class T> bool isDyadic (const ImageSet< T > & pyr)`

Check whether the pyramid is dyadic.

A dyadic pyramid is one in which each level is one half the width and one half the height of the preceding level.

Definition at line 320 of file Pyramid\_Ops.C.

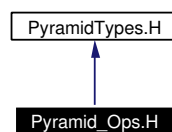
References `ImageSet< T >::getDepth()`, `Dims::h()`, `uint`, and `Dims::w()`.

Referenced by `makeImageArray()`, `SingleChannel::storePyramid()`, and `SingleChannel::storePyramid2()`.

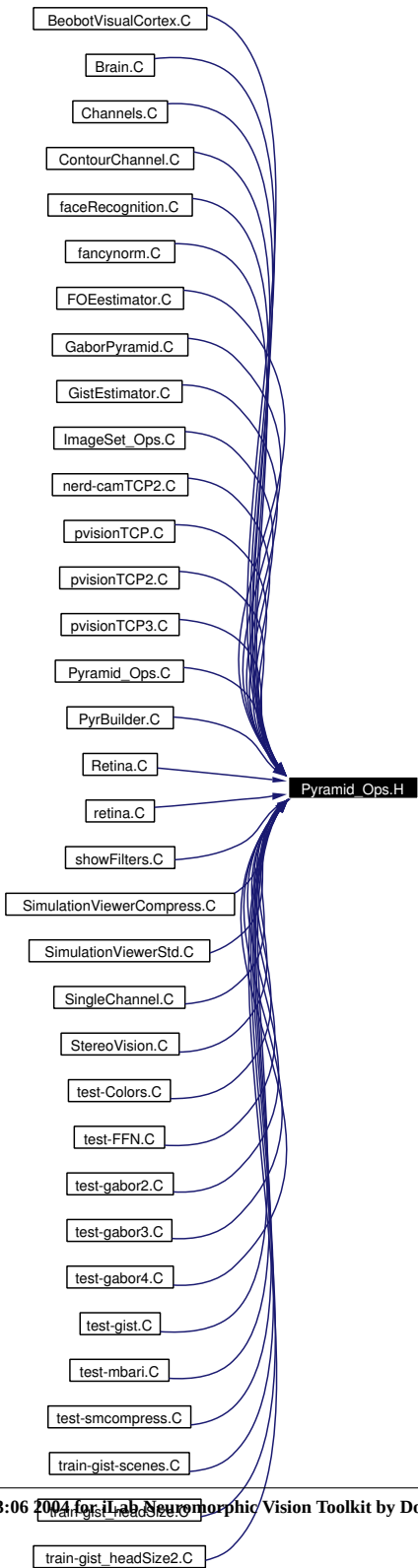
## 11.332 Pyramid\_Ops.H File Reference

```
#include "PyramidTypes.H"
```

Include dependency graph for Pyramid\_Ops.H:



This graph shows which files directly or indirectly include this file:



## Enumerations

- enum **GaborFlags** { **DO\_ENERGY\_NORM** = 1, **DO\_LAPLACIAN** = 2, **DO\_CLAMPED\_DIFF** = 4, **NO\_ABS** = 8 }

## Functions

- template<class T\_or\_RGB> [ImageSet](#)< T\_or\_RGB > [buildPyrGaussian](#) (const [Image](#)< T\_or\_RGB > &image, int depth, int filterSize)  
*Uses a Gaussian filter of size filterSize.*
- template<class T\_or\_RGB> [ImageSet](#)< T\_or\_RGB > [buildPyrConvolve](#) (const [Image](#)< T\_or\_RGB > &image, int depth, const [Image](#)< float > &filter)  
*Uses an arbitrary filter given as argument.*
- template<class T\_or\_RGB> [ImageSet](#)< T\_or\_RGB > [buildPyrLaplacian](#) (const [Image](#)< T\_or\_RGB > &image, int depth, int filterSize)  
*Uses a Laplacian filter of size filterSize.*
- template<class T\_or\_RGB> [ImageSet](#)< T\_or\_RGB > [buildPyrOriented](#) (const [Image](#)< T\_or\_RGB > &image, int depth, int filterSize, float theta, float intens=10.0f)  
*Uses an oriented Laplacian filter with a given size/orientation/intensity.*
- template<class T\_or\_RGB> [ImageSet](#)< T\_or\_RGB > [buildPyrLocalAvg](#) (const [Image](#)< T\_or\_RGB > &image, int depth)  
*Just fills the deepest level with the local image average at that scale.*
- template<class T\_or\_RGB> [ImageSet](#)< T\_or\_RGB > [buildPyrLocalMax](#) (const [Image](#)< T\_or\_RGB > &image, int depth)  
*Just fills the deepest level with the local image maximum at that scale.*
- template<class T\_or\_RGB> [ImageSet](#)< T\_or\_RGB > [buildPyrGeneric](#) (const [Image](#)< T\_or\_RGB > &image, int depth, const [PyramidType](#) typ, const float gabor\_theta=0.0f, const float gabor\_intens=10.0f)  
*Build a pyramid of type given by the PyramidType.*
- [ImageSet](#)< float > [buildPyrGabor](#) (const [ImageSet](#)< float > &gaussianPyr, float angle, float filter\_period, float elongation=1.0, int size=-1, int flags=0)  
*Build a Gabor Pyramid with a given angle, filter period and elongation.*
- [ImageSet](#)< float > [buildPyrGabor](#) (const [Image](#)< float > &img, int depth, float angle, float filter\_period, float elongation=1.0, int size=-1, int flags=0)



*Build a Gabor Pyramid with a given angle, filter period and elongation.*

- `template<class T> bool isDyadic (const ImageSet< T > &pyr)`  
*Check whether the pyramid is dyadic.*
- `template<class T> Image< T > centerSurround (const ImageSet< T > &pyr, const int lev1, const int lev2, const bool absol=false, const ImageSet< float > *clipPyr=0)`  
*Compute center-surround between images at lev1 and lev2.*
- `template<class T> void centerSurround (const ImageSet< T > &pyr, const int lev1, const int lev2, Image< T > &pos, Image< T > &neg, const ImageSet< float > *clipPyr=0)`  
*Compute center-surround between images, splitting positive and negative.*
- `template<class T> Image< T > centerSurroundSingleOpponent (const ImageSet< T > &cpyr, const ImageSet< T > &spyr, const int lev1, const int lev2, const bool absol=false, const ImageSet< float > *clipPyr=0)`  
*Compute single-opponent center-surround between images at lev1 and lev2.*
- `template<class T> void centerSurroundSingleOpponent (const ImageSet< T > &cpyr, const ImageSet< T > &spyr, const int lev1, const int lev2, Image< T > &pos, Image< T > &neg, const ImageSet< float > *clipPyr=0)`  
*Compute single-opp center-surr between imgs, splitting pos and neg.*
- `template<class T> Image< T > centerSurroundDiff (const ImageSet< T > &pyr1, const ImageSet< T > &pyr2, const int lev1, const int lev2, const bool absol=false, const ImageSet< float > *clipPyr=0)`  
*Compute center-surround between difference images at lev1 and lev2.*
- `template<class T> void centerSurroundDiff (const ImageSet< T > &pyr1, const ImageSet< T > &pyr2, const int lev1, const int lev2, Image< T > &pos, Image< T > &neg, const ImageSet< float > *clipPyr=0)`  
*Compute center-surround differences, splitting pos & neg values.*
- `template<class T> Image< T > centerSurroundDiffSingleOpponent (const ImageSet< T > &cpyr1, const ImageSet< T > &cpyr2, const ImageSet< T > &spyr1, const ImageSet< T > &spyr2, const int lev1, const int lev2, const bool absol=false, const ImageSet< float > *clipPyr=0)`  
*Compute single-opp center-surround btw difference images at lev1 and lev2.*
- `template<class T> void centerSurroundDiffSingleOpponent (const ImageSet< T > &cpyr1, const ImageSet< T > &cpyr2, const ImageSet< T > &spyr1, const ImageSet< T > &spyr2, const int lev1, const int lev2, Image< T > &pos, Image< T > &neg, const ImageSet< float > *clipPyr=0)`

*Compute single-opp center-surround differnces, splitting pos & neg values.*

### 11.332.1 Detailed Description

Free functions operating on pyramid data structures

Definition in file [Pyramid\\_Ops.H](#).

### 11.332.2 Function Documentation

**11.332.2.1** `template<class T_or_RGB> ImageSet<T_or_RGB>  
buildPyrConvolve (const Image< T_or_RGB > & image, int depth,  
const Image< float > & filter)`

Uses an arbitrary filter given as argument.

Referenced by `RGBConvolvePyrBuilder< T >::build()`, `ConvolvePyrBuilder< T >::build()`, and `RGBConvolvePyrBuilder< T >::build2()`.

**11.332.2.2** `ImageSet<float> buildPyrGabor (const Image< float > & img, int  
depth, float angle, float filter_period, float elongation = 1.0, int size =  
-1, int flags = 0)`

Build a Gabor Pyramid with a given angle, filter period and elongation.

**Parameters:**

*size* determines the filter size (in pixels) - in the default case (-1), the size is computed from the filter\_preiod and elongation

Definition at line 295 of file `Pyramid_Ops.C`.

References `buildPyrGaussian()`, and `buildPyrLaplacian()`.

Referenced by `GaborPyrBuilder< T >::build()`, and `GaborPyrBuilder< T >::input()`.

**11.332.2.3** `ImageSet<float> buildPyrGabor (const ImageSet< float > &  
gaussianPyr, float angle, float filter_period, float elongation = 1.0, int  
size = -1, int flags = 0)`

Build a Gabor Pyramid with a given angle, filter period and elongation.

**Parameters:**

*size* determines the filter size (in pixels) - in the default case (-1), the size is computed from the filter\_preiod and elongation

Definition at line 225 of file Pyramid\_Ops.C.

References `abs()`, `energyNorm()`, `gaborFilter3()`, `ImageSet< T >::getDepth()`, `Image< T >::getHeight()`, `Image< T >::getWidth()`, `optConvolve()`, and `uint`.

**11.332.2.4** `template<class T_or_RGB> ImageSet<T_or_RGB>  
buildPyrGaussian (const Image< T_or_RGB > & image, int depth,  
int filterSize)`

Uses a Gaussian filter of size *filterSize*.

Referenced by `GaussianPyrBuilder< T >::build()`, `buildPyrGabor()`, `RetinaStd::doGetOutput()`, `StereoChannel::doInput()`, `DisparityChannel::doInput()`, `main()`, `maxNormalizeFancyFast()`, `SimulationViewerStd::newInputFrame()`, `SimulationViewerCompress::newInputFrame()`, `SingleChannel::setClipPyramid()`, `BeobotVisualCortex::singleCPUprocess()`, and `FOEestimator::updateFOE()`.

**11.332.2.5** `template<class T_or_RGB> ImageSet<T_or_RGB>  
buildPyrGeneric (const Image< T_or_RGB > & image, int depth,  
const PyramidType typ, const float gabor_theta = 0.0f, const float  
gabor_intens = 10.0f)`

Build a pyramid of type given by the `PyramidType`.

Referenced by `ReichardtPyrBuilder< T >::build()`, `GenericPyrBuilder< T >::build()`, and `computeCMAP()`.

**11.332.2.6** `template<class T_or_RGB> ImageSet<T_or_RGB>  
buildPyrLaplacian (const Image< T_or_RGB > & image, int depth,  
int filterSize)`

Uses a Laplacian filter of size *filterSize*.

Referenced by `LaplacianPyrBuilder< T >::build()`, and `buildPyrGabor()`.

**11.332.2.7** `template<class T_or_RGB> ImageSet<T_or_RGB>  
buildPyrLocalAvg (const Image< T_or_RGB > & image, int depth)`

Just fills the deepest level with the local image average at that scale.

**11.332.2.8** `template<class T_or_RGB> ImageSet<T_or_RGB>  
buildPyrLocalMax (const Image< T_or_RGB > & image, int depth)`

Just fills the deepest level with the local image maximum at that scale.

**11.332.2.9** `template<class T_or_RGB> ImageSet<T_or_RGB>  
buildPyrOriented (const Image< T_or_RGB > & image, int depth,  
int filterSize, float theta, float intens = 10.0f)`

Uses an oriented Laplacian filter with a given size/orientation/intensity.

This is a fast approximation to gabor convolution.

Referenced by OrientedPyrBuilder< T >::build(), and BeobotVisualCortex::single-CPUprocess().

**11.332.2.10** `template<class T> void centerSurround (const ImageSet< T > &  
pyr, const int lev1, const int lev2, Image< T > & pos, Image< T >  
& neg, const ImageSet< float > * clipPyr = 0)`

Compute center-surround between images, splitting positive and negative.

Definition at line 369 of file Pyramid\_Ops.C.

References ASSERT, ImageSet< T >::getDepth(), ImageSet< T >::isNonEmpty(), and uint.

Referenced by centerSurround(), centerSurroundDiff(), centerSurroundDiffSingleOpponent(), centerSurroundSingleOpponent(), computeCMap(), and doubleOpp().

**11.332.2.11** `template<class T> Image<T> centerSurround (const ImageSet<  
T > & pyr, const int lev1, const int lev2, const bool absol = false,  
const ImageSet< float > * clipPyr = 0)`

Compute center-surround between images at lev1 and lev2.

Definition at line 342 of file Pyramid\_Ops.C.

References ASSERT, centerSurround(), ImageSet< T >::getDepth(), ImageSet< T >::isNonEmpty(), and uint.

**11.332.2.12** `template<class T> void centerSurroundDiff (const ImageSet< T >  
& pyr1, const ImageSet< T > & pyr2, const int lev1, const int lev2,  
Image< T > & pos, Image< T > & neg, const ImageSet< float > *  
clipPyr = 0)`

Compute center-surround differences, splitting pos & neg values.

Definition at line 490 of file Pyramid\_Ops.C.

References ASSERT, centerSurround(), ImageSet< T >::getDepth(), Image< T >::getDims(), ImageSet< T >::isNonEmpty(), and uint.

**11.332.2.13** `template<class T> Image<T> centerSurroundDiff (const ImageSet< T > & pyr1, const ImageSet< T > & pyr2, const int lev1, const int lev2, const bool absol = false, const ImageSet< float > * clipPyr = 0)`

Compute center-surround between difference images at lev1 and lev2.

Difference images are obtained at each level by taking the difference between the image from \*this minus the image from other\_pyr

Definition at line 456 of file Pyramid\_Ops.C.

References ASSERT, centerSurround(), **ImageSet**< T >::getDepth(), **Image**< T >::getDims(), **ImageSet**< T >::isNonEmpty(), and uint.

**11.332.2.14** `template<class T> void centerSurroundDiffSingleOpponent (const ImageSet< T > & cpyr1, const ImageSet< T > & cpyr2, const ImageSet< T > & spyr1, const ImageSet< T > & spyr2, const int lev1, const int lev2, Image< T > & pos, Image< T > & neg, const ImageSet< float > * clipPyr = 0)`

Compute single-opp center-surround differnces, splitting pos & neg values.

Definition at line 562 of file Pyramid\_Ops.C.

References ASSERT, centerSurround(), **ImageSet**< T >::getDepth(), **Image**< T >::getDims(), **ImageSet**< T >::isNonEmpty(), and uint.

**11.332.2.15** `template<class T> Image<T> centerSurroundDiffSingleOpponent (const ImageSet< T > & cpyr1, const ImageSet< T > & cpyr2, const ImageSet< T > & spyr1, const ImageSet< T > & spyr2, const int lev1, const int lev2, const bool absol = false, const ImageSet< float > * clipPyr = 0)`

Compute single-opp center-surround btw difference images at lev1 and lev2.

Difference images are obtained at each level by taking the difference between the image from \*this minus the image from other\_pyr

Definition at line 524 of file Pyramid\_Ops.C.

References ASSERT, centerSurround(), **ImageSet**< T >::getDepth(), **Image**< T >::getDims(), **ImageSet**< T >::isNonEmpty(), and uint.

**11.332.2.16** `template<class T> void centerSurroundSingleOpponent (const ImageSet< T > & cpyr, const ImageSet< T > & spyr, const int lev1, const int lev2, Image< T > & pos, Image< T > & neg, const ImageSet< float > * clipPyr = 0)`

Compute single-opp center-surr between imgs, splitting pos and neg.

Definition at line 426 of file Pyramid\_Ops.C.

References ASSERT, centerSurround(), ImageSet< T >::getDepth(), ImageSet< T >::isNonEmpty(), and uint.

**11.332.2.17** `template<class T> Image<T> centerSurroundSingleOpponent (const ImageSet< T > & cpyr, const ImageSet< T > & spyr, const int lev1, const int lev2, const bool absol = false, const ImageSet< float > * clipPyr = 0)`

Compute single-opponent center-surround between images at lev1 and lev2.

This is essentially cpyr[lev1] - spyr[lev2]

Definition at line 395 of file Pyramid\_Ops.C.

References ASSERT, centerSurround(), ImageSet< T >::getDepth(), ImageSet< T >::isNonEmpty(), and uint.

**11.332.2.18** `template<class T> bool isDyadic (const ImageSet< T > & pyr)`

Check whether the pyramid is dyadic.

A dyadic pyramid is one in which each level is one half the width and one half the height of the preceding level.

Definition at line 320 of file Pyramid\_Ops.C.

References ImageSet< T >::getDepth(), Dims::h(), uint, and Dims::w().

Referenced by makeImageArray(), SingleChannel::storePyramid(), and SingleChannel::storePyramid2().

## 11.333 PyramidTypes.H File Reference

This graph shows which files directly or indirectly include this file:

### Defines

- `#define` [NBPYRAMIDTYPES](#) 11  
*number of known PyramidTypes:*

### Enumerations

- `enum` [PyramidType](#) {  
    [Gaussian3](#) = 0, [Gaussian5](#) = 1, [Gaussian9](#) = 2, [Laplacian5](#) = 3,  
    [Laplacian9](#) = 4, [Oriented5](#) = 5, [Oriented9](#) = 6, [QuickLocalAvg](#) = 7,  
    [QuickLocalMax](#) = 8, [BlockScale](#) = 9, [Gabor9](#) = 10 }  
*Possible types of pyramids.*

### Functions

- `const char *` [pyramidTypeName](#) (`const` [PyramidType](#) p)

#### 11.333.1 Detailed Description

Definition of the pyramid types

Definition in file [PyramidTypes.H](#).

#### 11.333.2 Define Documentation

##### 11.333.2.1 `#define` NBPYRAMIDTYPES 11

number of known PyramidTypes:

Definition at line 58 of file [PyramidTypes.H](#).

Referenced by [convertFromString\(\)](#).

### 11.333.3 Enumeration Type Documentation

#### 11.333.3.1 enum [PyramidType](#)

Possible types of pyramids.

**Enumeration values:**

***Gaussian3*** Corresponds to [GaussianPyrBuilder](#) with filter width 3.

***Gaussian5*** Corresponds to [GaussianPyrBuilder](#) with filter width 5.

***Gaussian9*** Corresponds to [GaussianPyrBuilder](#) with filter width 9.

***Laplacian5*** Corresponds to [LaplacianPyrBuilder](#) with filter width 5.

***Laplacian9*** Corresponds to [LaplacianPyrBuilder](#) with filter width 9.

***Oriented5*** Corresponds to [OrientedPyrBuilder](#) with filter width 5.

***Oriented9*** Corresponds to [OrientedPyrBuilder](#) with filter width 9.

Definition at line 42 of file PyramidTypes.H.

Referenced by [convertFromString\(\)](#).



## 11.334 PyrBuilder.C File Reference

```
#include "PyrBuilder.H"
#include "Image.H"
#include "ImageSet.H"
#include "Pyramid_Ops.H"
#include "Image_FilterOps.H"
#include "Image_ColorOps.H"
#include "Image_ShapeOps.H"
#include "Image_MathOps.H"
```

Include dependency graph for PyrBuilder.C:

### Defines

- `#define INSTANTIATE(T)`

### 11.334.1 Detailed Description

Classes for building dyadic pyramids

Definition in file [PyrBuilder.C](#).

### 11.334.2 Define Documentation

#### 11.334.2.1 `#define INSTANTIATE(T)`

**Value:**

```
template class PyrBuilder< T >; \
template class GenericPyrBuilder< T >; \
template class GaussianPyrBuilder< T >; \
template class ConvolvePyrBuilder< T >; \
template class LaplacianPyrBuilder< T >; \
template class OrientedPyrBuilder< T >; \
template class ReichardtPyrBuilder< T >; \
```

Definition at line 379 of file PyrBuilder.C.

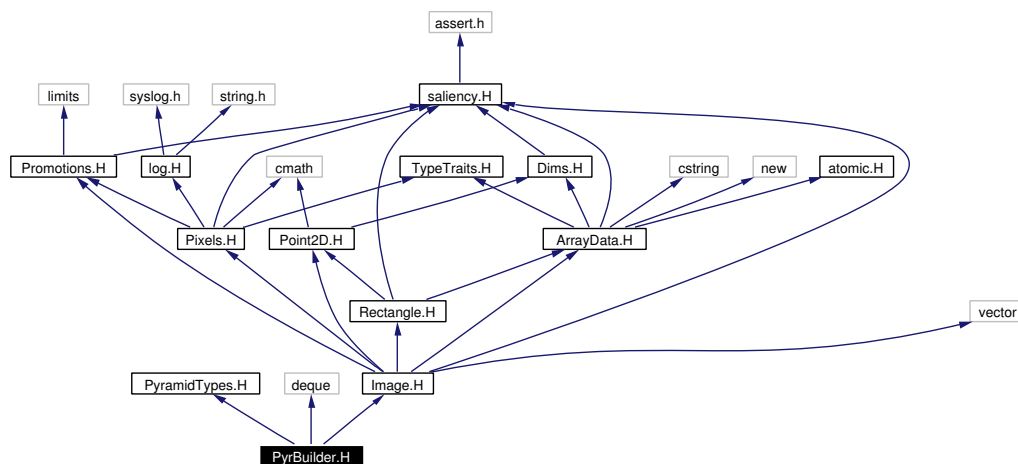
## 11.335 PyrBuilder.H File Reference

```
#include "PyramidTypes.H"
```

```
#include <deque>
```

```
#include "Image.H"
```

Include dependency graph for PyrBuilder.H:



This graph shows which files directly or indirectly include this file:

### Classes

- class [PyrBuilder](#)  
*An interface class for creating dyadic pyramids from input images.*
- class [GaussianPyrBuilder](#)  
*Builds pyramids based on Gaussian filters.*
- class [ConvolvePyrBuilder](#)  
*Builds pyramids based on arbitrary filters.*
- class [RGBConvolvePyrBuilder](#)  
*Builds pyramids based on arbitrary color filters.*
- class [LaplacianPyrBuilder](#)

*Builds pyramids based on Laplacian filters.*

- class [OrientedPyrBuilder](#)  
*Builds pyramid based on oriented Laplacian filters.*
- class [GenericPyrBuilder](#)  
*Builds pyramids of a type specified by a PyramidType.*
- class [ReichardtPyrBuilder](#)  
*This class implements Reichardt motion detections.*
- class [TemplateMatchPyrBuilder](#)  
*A [PyrBuilder](#) that does template matching.*
- class [GaborPyrBuilder](#)  
*A pyramid based on convolution with oriented Gabor filters.*

### 11.335.1 Detailed Description

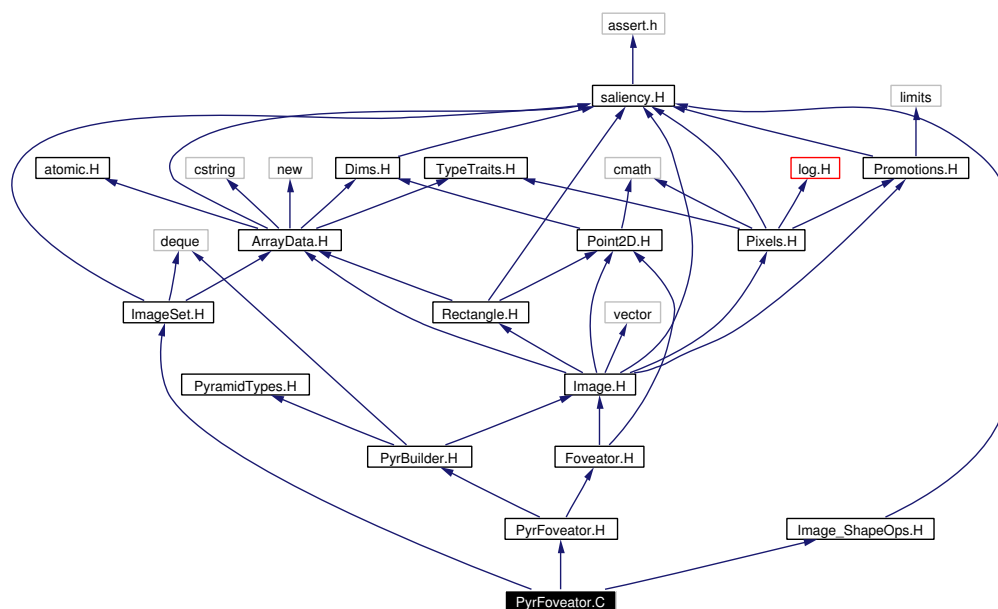
Classes for building dyadic pyramids

Definition in file [PyrBuilder.H](#).

## 11.336 PyrFoveator.C File Reference

```
#include "PyrFoveator.H"
#include "ImageSet.H"
#include "Image_ShapeOps.H"
```

Include dependency graph for PyrFoveator.C:



### 11.336.1 Detailed Description

**Foveator** class that builds dyadic pyramid

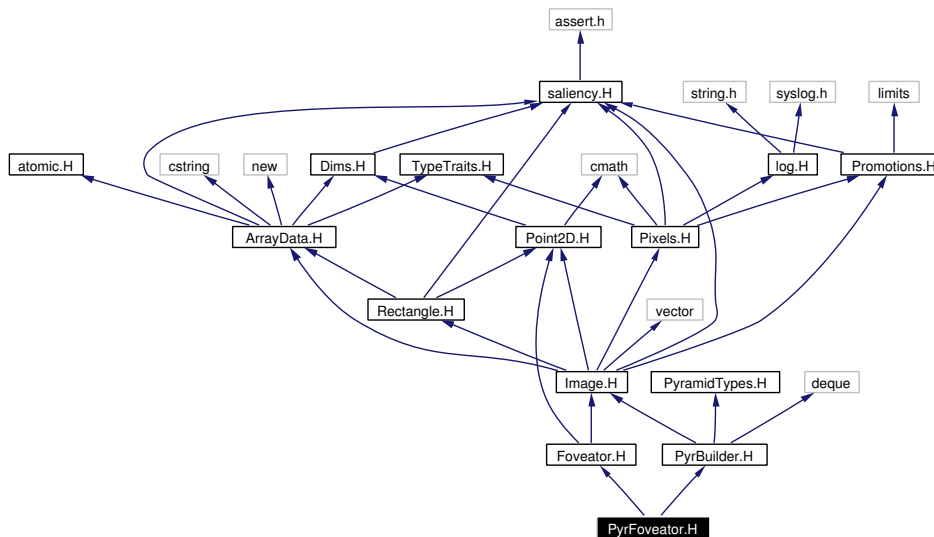
Definition in file [PyrFoveator.C](#).

## 11.337 PyrFoveator.H File Reference

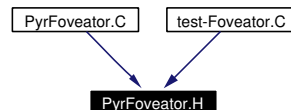
```
#include "Foveator.H"
```

```
#include "PyrBuilder.H"
```

Include dependency graph for PyrFoveator.H:



This graph shows which files directly or indirectly include this file:



### Classes

- class [PyrFoveator](#)

*A class that does space-variant processing with dyadic pyramid.*

### 11.337.1 Detailed Description

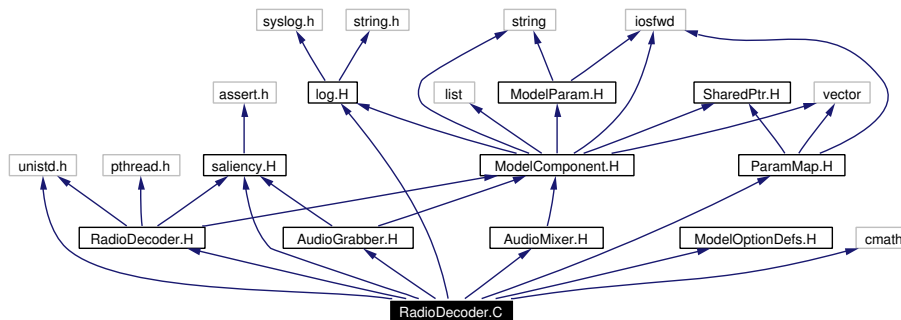
[Foveator](#) class that builds dyadic pyramid

Definition in file [PyrFoveator.H](#).

## 11.338 RadioDecoder.C File Reference

```
#include <unistd.h>
#include "saliency.H"
#include "log.H"
#include "RadioDecoder.H"
#include "AudioMixer.H"
#include "AudioGrabber.H"
#include "ParamMap.H"
#include "ModelOptionDefs.H"
#include <cmath>
```

Include dependency graph for RadioDecoder.C:



### Defines

- `#define RD_WIN 2`
- `#define RD_THRESH (255 - 20)`
- `#define MINRDPULSE 30`
- `#define MAXRDPULSE 150`

### Functions

- `void * RadioDecoder_run (void *r0)`

### 11.338.1 Detailed Description

Decode radio pulse-width-modulated signals

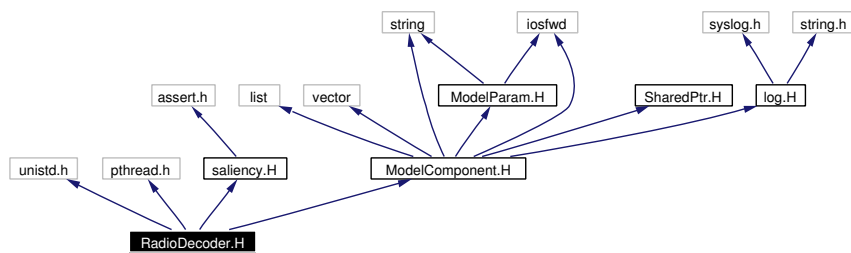
Definition in file [RadioDecoder.C](#).



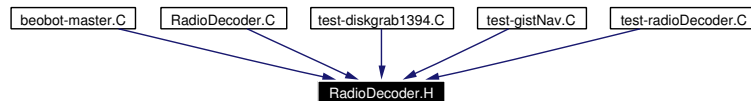
## 11.339 RadioDecoder.H File Reference

```
#include <unistd.h>
#include <pthread.h>
#include "saliency.H"
#include "ModelComponent.H"
```

Include dependency graph for RadioDecoder.H:



This graph shows which files directly or indirectly include this file:



### Classes

- class [RadioDecoder](#)

*This class runs a thread which records servo position from sound card.*

### 11.339.1 Detailed Description

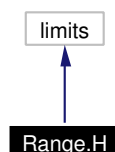
Decode radio pulse-width-modulated signals

Definition in file [RadioDecoder.H](#).

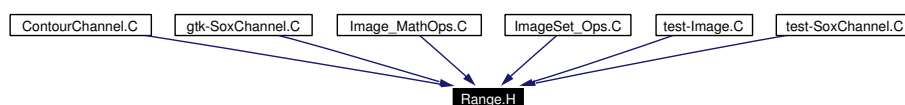
## 11.340 Range.H File Reference

```
#include <limits>
```

Include dependency graph for Range.H:



This graph shows which files directly or indirectly include this file:



### Classes

- class [Range](#)  
*A class for managing min/max range information.*

### 11.340.1 Detailed Description

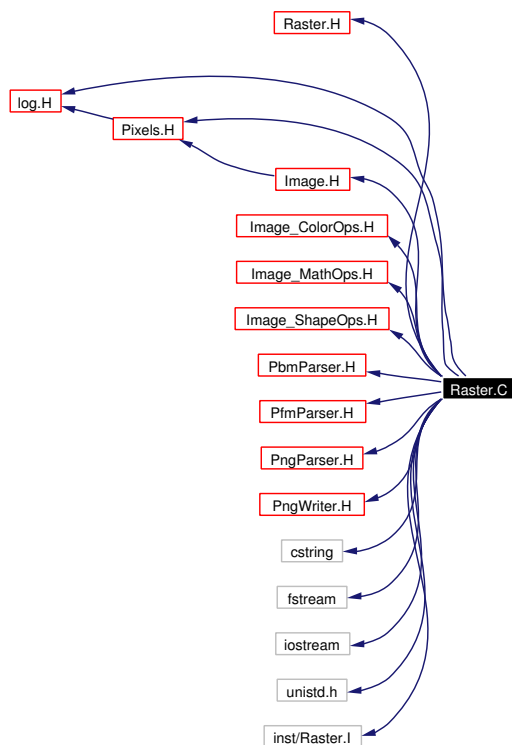
a class to manage min/max range information.

Definition in file [Range.H](#).

## 11.341 Raster.C File Reference

```
#include "Raster.H"
#include "Image.H"
#include "Image_ColorOps.H"
#include "Image_MathOps.H"
#include "Image_ShapeOps.H"
#include "log.H"
#include "Pixels.H"
#include "PbmParser.H"
#include "PfmParser.H"
#include "PngParser.H"
#include "PngWriter.H"
#include <cstring>
#include <fstream>
#include <iostream>
#include <unistd.h>
#include "inst/Raster.I"
```

Include dependency graph for Raster.C:



## Defines

- `#define INST_NAMESPACE Raster`

## Functions

- `RasterFileFormat AddFileExtension` (char \*file, const `RasterFileFormat` ftype, const `Raster::ColorMode` mode)
- void `waitDelete` (const char \*fname)
- void `writeGrayA` (const `Image< byte >` &image, `RasterFileFormat` ftype, const char \*fname, va\_list ap)
- void `writeFloatA` (const `Image< float >` &image, `RasterFileFormat` ftype, const char \*fname, va\_list ap)
- `Image< byte >` `readGrayA` (`RasterFileFormat` ftype, const char \*fname, va\_list ap)
- `Image< float >` `readFloatA` (`RasterFileFormat` ftype, const char \*fname, va\_list ap)
- void `writeRGBA` (const `Image< PixRGB< byte > >` &image, `RasterFileFormat` ftype, const char \*fname, va\_list ap)

- `Image< PixRGB< byte > > readRGBA (RasterFileFormat ftype, const char *fname, va_list ap)`
- `template<class T> void doVisu (const Image< T > &image, const RasterFileFormat ftype, const bool normalize, const bool waitdel, const char *fname, va_list ap)`
- `template<class T> void doVisuRGB (const Image< PixRGB< T > > &image, const RasterFileFormat ftype, const bool normalize, const bool waitdel, const char *fname, va_list ap)`

## Variables

- `const int FNAMELEN = 1024`
- `bool USING_XV = true`

### 11.341.1 Detailed Description

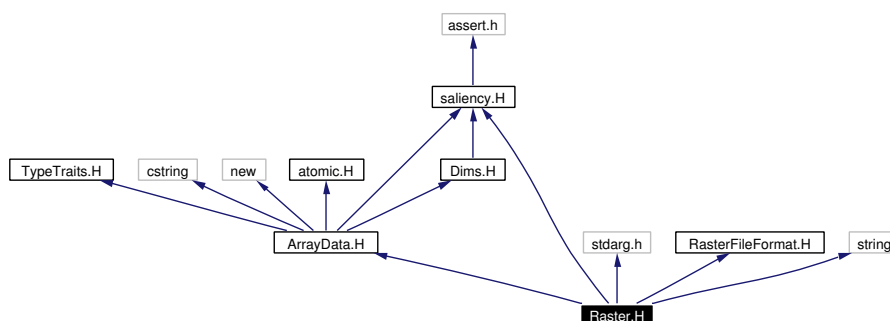
Read/write/display raster images

Definition in file [Raster.C](#).

## 11.342 Raster.H File Reference

```
#include "saliency.H"  
#include <stdarg.h>  
#include "RasterFileFormat.H"  
#include "ArrayData.H"  
#include <string>
```

Include dependency graph for Raster.H:



This graph shows which files directly or indirectly include this file:

### Namespaces

- namespace [Raster](#)

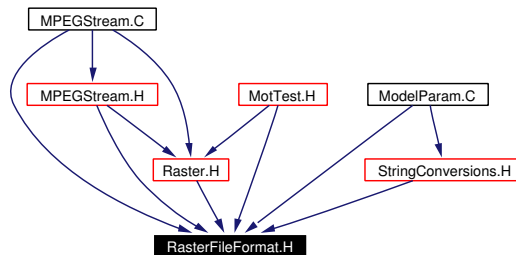
### 11.342.1 Detailed Description

Writes a raster image to disk / reads it / displays it

Definition in file [Raster.H](#).

## 11.343 RasterFileFormat.H File Reference

This graph shows which files directly or indirectly include this file:



### Enumerations

- enum [RasterFileFormat](#) {  
**PNM** = 0, **PNG**, **PFM**, **RAWIMAGE**,  
**Auto** }

*Definition of supported [Raster](#) file formats.*

### 11.343.1 Detailed Description

file formats supported by [Raster](#)

Definition in file [RasterFileFormat.H](#).

### 11.343.2 Enumeration Type Documentation

#### 11.343.2.1 enum [RasterFileFormat](#)

Definition of supported [Raster](#) file formats.

See [Raster.H](#) for functions using these file formats. If you add new formats here, be sure to also update [StringConversions.C](#)

Definition at line 44 of file RasterFileFormat.H.

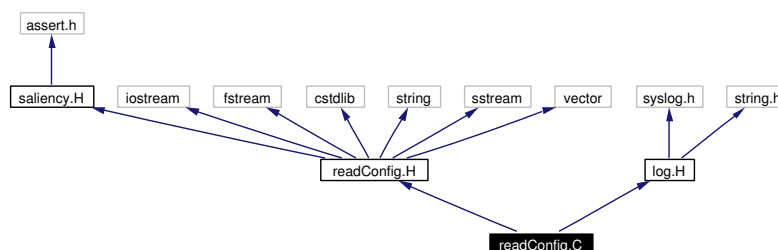
Referenced by `FrameSeries::FrameSeries()`, `Raster::getImageDims()`, `Raster::getImageMeta()`, `InputFrameSeries::readGray()`, `InputFrameSeries::readRGB()`, `OutputFrameSeries::writeFloat()`, `OutputFrameSeries::writeGray()`, and `OutputFrameSeries::writeRGB()`.

## 11.344 readConfig.C File Reference

```
#include "readConfig.H"
```

```
#include "log.H"
```

Include dependency graph for readConfig.C:



### 11.344.1 Detailed Description

[CINNIC](#) classes

Definition in file [readConfig.C](#).



## 11.345 readConfig.H File Reference

```
#include "saliency.H"
#include <iostream>
#include <fstream>
#include <cstdlib>
#include <string>
#include <sstream>
#include <vector>
```

Include dependency graph for readConfig.H:

This graph shows which files directly or indirectly include this file:

### Classes

- class [readConfig](#)  
*open a config file and parse it for data retrieval*

### 11.345.1 Detailed Description

[CINNIC](#) classes

Definition in file [readConfig.H](#).

## 11.346 Rectangle.H File Reference

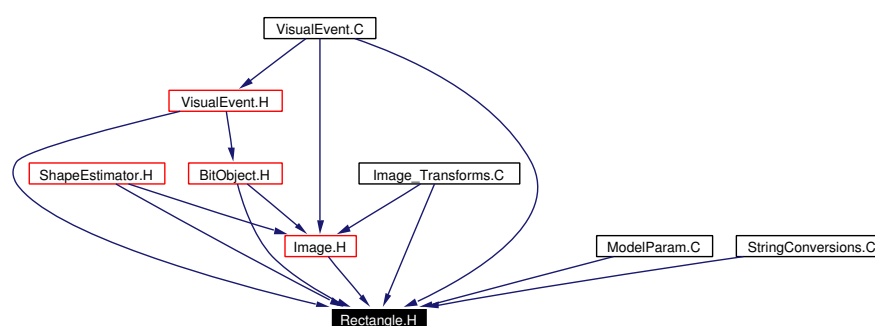
```
#include "saliency.H"
```

```
#include "ArrayData.H"
```

```
#include "Point2D.H"
```

Include dependency graph for Rectangle.H:

This graph shows which files directly or indirectly include this file:



### Classes

- class [Rectangle](#)  
*A basic rectangle class.*

### 11.346.1 Detailed Description

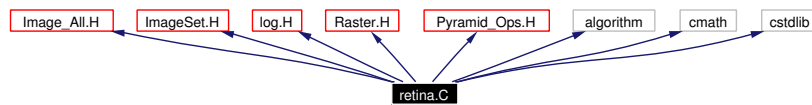
A basic rectangle class

Definition in file [Rectangle.H](#).

## 11.347 retina.C File Reference

```
#include "Image_All.H"  
#include "ImageSet.H"  
#include "log.H"  
#include "Raster.H"  
#include "Pyramid_Ops.H"  
#include <algorithm>  
#include <cmath>  
#include <cstdlib>
```

Include dependency graph for retina.C:



### Defines

- #define **PYR\_DEPTH** 10  
*depth of the pyramid used to obtain the eccentricity-dependent resolution*
- #define **BLIND\_X** 0.15f  
*blind spot center, as a factor of image width and height, from center*
- #define **BLIND\_Y** 0.00f
- #define **BLIND\_2S2** 600.0f  
*blind spot  $2 \times \sigma^2$ , in pixels*
- #define **FOVEA\_2S2** 150.0f  
*fovea  $2 \times \sigma^2$ , in pixels:*

### Functions

- int **main** (int argc, char \*\*argv)

### 11.347.1 Detailed Description

Apply a retinalike transformation to an image

Definition in file [retina.C](#).

### 11.347.2 Define Documentation

#### 11.347.2.1 `#define BLIND_2S2 600.0f`

blind spot  $2 \cdot \sigma^2$ , in pixels

Definition at line 54 of file retina.C.

#### 11.347.2.2 `#define BLIND_X 0.15f`

blind spot center, as a factor of image width and height, from center

Definition at line 50 of file retina.C.

#### 11.347.2.3 `#define FOVEA_2S2 150.0f`

fovea  $2 \cdot \sigma^2$ , in pixels:

Definition at line 57 of file retina.C.

#### 11.347.2.4 `#define PYR_DEPTH 10`

depth of the pyramid used to obtain the eccentricity-dependent resolution

Definition at line 47 of file retina.C.

## 11.348 Retina.C File Reference

```
#include "Retina.H"
#include "ModelManager.H"
#include "ModelOptionDefs.H"
#include "Raster.H"
#include "Image_ShapeOps.H"
#include "SaccadeController.H"
#include "Brain.H"
#include "Image_Transforms.H"
#include "Pyramid_Ops.H"
```

Include dependency graph for Retina.C:

### 11.348.1 Detailed Description

a human retina

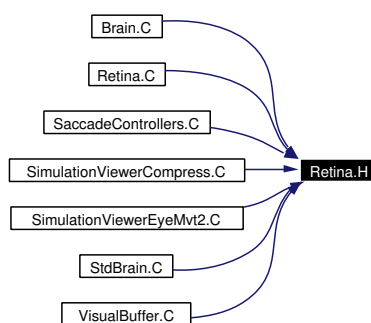
Definition in file [Retina.C](#).

## 11.349 Retina.H File Reference

```
#include "saliency.H"
#include "ModelComponent.H"
#include "Point2D.H"
#include "Image.H"
```

Include dependency graph for Retina.H:

This graph shows which files directly or indirectly include this file:



### Classes

- class [Retina](#)  
*The retina base class.*
- class [RetinaConfigurator](#)  
*Retina configurator.*
- class [RetinaStd](#)  
*Standard Retina implementation.*

### 11.349.1 Detailed Description

simulation of a human retina

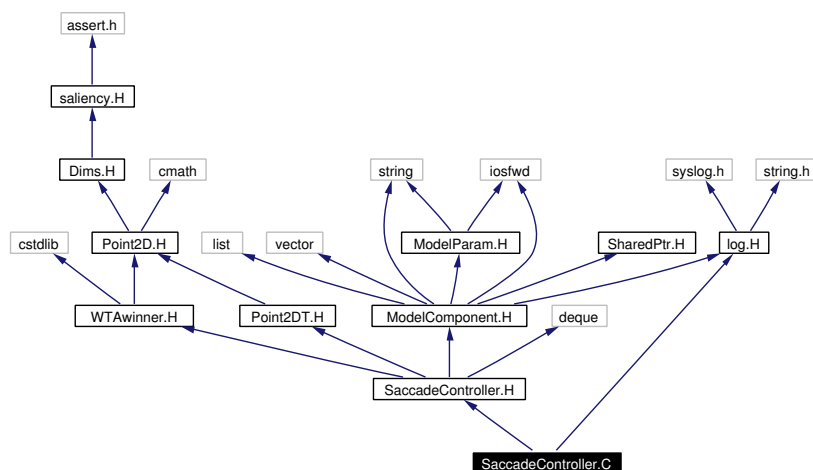
Definition in file [Retina.H](#).

## 11.350 SaccadeController.C File Reference

```
#include "SaccadeController.H"
```

```
#include "log.H"
```

Include dependency graph for SaccadeController.C:



### 11.350.1 Detailed Description

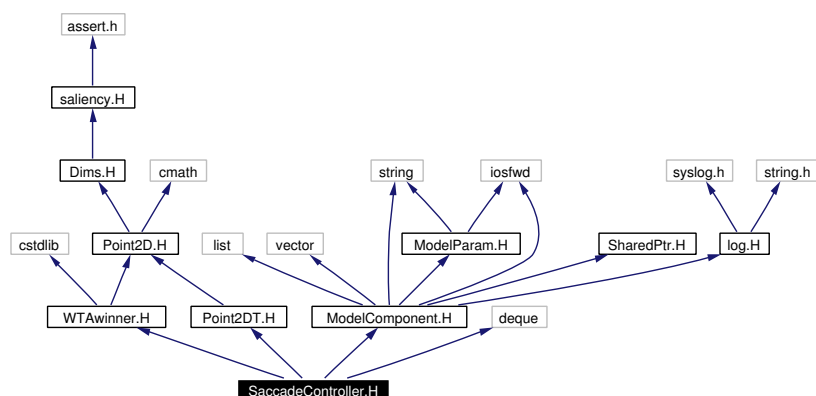
Base class for saccade generation

Definition in file [SaccadeController.C](#).

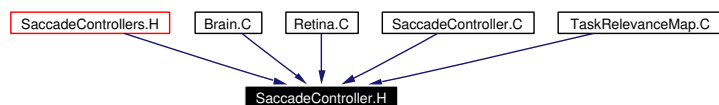
## 11.351 SaccadeController.H File Reference

```
#include "Point2DT.H"
#include "WTAWinner.H"
#include "ModelComponent.H"
#include <deque>
```

Include dependency graph for SaccadeController.H:



This graph shows which files directly or indirectly include this file:



### Classes

- class [SaccadeController](#)

*A base class to implement saccade controllers.*

### 11.351.1 Detailed Description

Base class for saccade generation

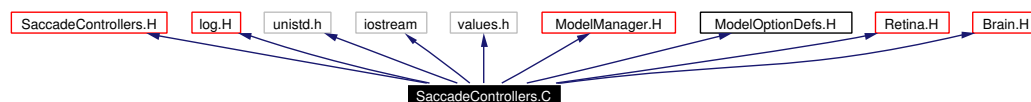
Definition in file [SaccadeController.H](#).



## 11.352 SaccadeControllers.C File Reference

```
#include "SaccadeControllers.H"  
#include "log.H"  
#include <unistd.h>  
#include <iostream>  
#include <values.h>  
#include "ModelManager.H"  
#include "ModelOptionDefs.H"  
#include "Retina.H"  
#include "Brain.H"
```

Include dependency graph for SaccadeControllers.C:



### Defines

- #define **DAMPED\_HARMONIC**(a, k, x) ((expf(-k\*x))\*(pow(sin(M\_PI/2.0\*x), a)))

### 11.352.1 Detailed Description

Derived classes for saccade generation

Definition in file [SaccadeControllers.C](#).

## 11.353 SaccadeControllers.H File Reference

```
#include "SaccadeController.H"
```

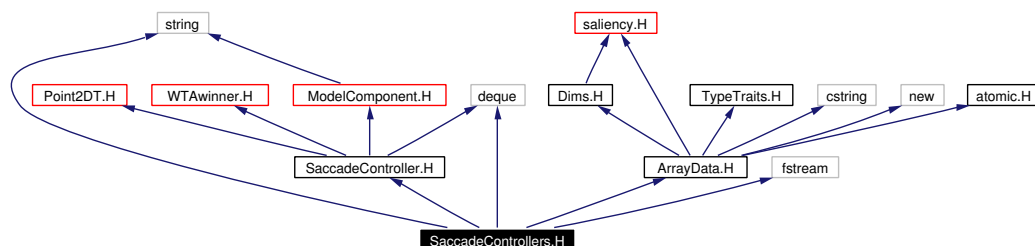
```
#include "ArrayData.H"
```

```
#include <fstream>
```

```
#include <string>
```

```
#include <deque>
```

Include dependency graph for SaccadeControllers.H:



This graph shows which files directly or indirectly include this file:

### Classes

- class [SaccadeControllerConfigurator](#)  
*SaccadeController configurator.*
- class [TrivialSaccadeController](#)  
*Trivial saccade controller.*
- class [FixedSaccadeController](#)  
*Fixed saccade controller.*
- class [FrictionSaccadeController](#)  
*Friction-damped mass-spring saccade controller.*
- class [ThresholdSaccadeController](#)  
*Threshold-limited controller that eliminates small or unreliable shifts.*
- class [ThresholdFrictionSaccadeController](#)  
*Threshold-limited friction-damped controller.*

- class [SimpleEyeHeadSaccadeController](#)  
*Simple eye/head controller.*
- class [MonkeyEyeHeadController](#)  
*Realistic Monkey Eye/Head controller.*
- class [Monkey2SaccadeController](#)  
*Realistic rhesus monkey saccade controller.*
- class [MonkeySaccadeController](#)  
*Realistic rhesus monkey saccade controller.*
- class **MonkeySaccadeController::SomeInfo**
- class **MonkeySaccadeController::BlinkInfo**
- class [EyeTrackerSaccadeController](#)  
*EyeTracker saccade controller.*
- class [VisBufSaccadeController](#)  
*VisBuf saccade controller.*
- class [ModelEyeHeadController](#)  
*Realistic Eye/Head controller.*
- class [ModelWrapperSaccadeController](#)  
*Realistic rhesus monkey saccade controller.*

### 11.353.1 Detailed Description

Derived classes for saccade generation

Definition in file [SaccadeControllers.H](#).

## 11.354 saccades.C File Reference

```
#include "Image.H"
#include "Image_ShapeOps.H"
#include "fancynorm.H"
#include "log.H"
#include "V4Lgrabber.H"
#include "VCC4.H"
#include "StdBrain.H"
#include "XWinManaged.H"
#include "ModelManager.H"
#include "ShapeEstimator.H"
#include "SaccadeControllers.H"
#include "SimulationViewerStd.H"
#include "VisualCortex.H"
#include "ChannelBase.H"
#include <algorithm>
#include <cmath>
#include <stdio.h>
#include <iostream>
```

Include dependency graph for saccades.C:

### Defines

- #define **CAM\_WINDOW\_NAME** "Saliency Calculated"
- #define **CAM\_PREV\_NAME** "Captured [Image](#)"
- #define **ANG\_FACTOR** (180.0/Pi)
- #define **MAXPAN** ((float)100.0)
- #define **MINPAN** ((float)-100.0)
- #define **MAXTILT** ((float)30.0)
- #define **MINTILT** ((float)-30.0)
- #define **TOO\_MUCH\_TIME** 0.7

## Functions

- float **sqr** (float x)
- float **randang** ()
- int **main** (const int argc, const char \*\*argv)

### 11.354.1 Detailed Description

Use a pan-tilt camera for executing saccades

Definition in file [saccades.C](#).

### 11.354.2 Function Documentation

#### 11.354.2.1 int main (const int *argc*, const char \*\* *argv*)

This program takes input from a camera, calculates the most salient spot(s) and moves the camera there according to the following rules:

1) take image from camera 2) calculate next most salient spot 3) If (spot is already in list) goto 2 4) store spot in list 5) move to the spot 6) goto 1

The list is finite (small) sized and "forgets" the oldest entries. When the search for the next most salient spot in a given image takes too much time, the camera is moved by a random amount.

Definition at line 103 of file saccades.C.

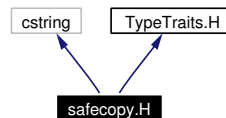
References `Image< T >::drawDisk()`, `XWinManaged::drawImage()`, `FULLTRACE`, `SharedPtr< T >::get()`, `Point2D::i`, `Point2D::j`, `main()`, `XWinManaged::pressedCloseButton()`, `rescale()`, `SharedPtr< T >::reset()`, and `XWindow::setTitle()`.

## 11.355 safecopy.H File Reference

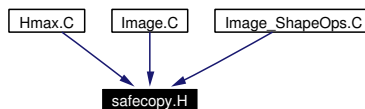
```
#include <cstring>
```

```
#include "TypeTraits.H"
```

Include dependency graph for safecopy.H:



This graph shows which files directly or indirectly include this file:



### Classes

- struct [SafeCopyHelper](#)  
*Helper template struct for pointer version of `safecopy()`.*
- struct [SafeCopyHelper< T, true >](#)  
*Specialized implementation for trivial types.*

### Functions

- template<class Iter, class ConstIter> void [genericcopy](#) (Iter dest, ConstIter src, unsigned int count)  
*Element-by-element copy a series forward iterators.*
- template<class Iter, class ConstIter> void [safecopy](#) (Iter dest, ConstIter src, unsigned int count)  
*A generic version of `safecopy()` for iterators.*
- template<class T> void [safecopy](#) (T \*dest, const T \*src, unsigned int count)  
*A specialized version of `safecopy()` for pointers.*

### 11.355.1 Detailed Description

Provides a "safe" replacement for memcpy: when MEM\_DEBUG is not defined, we just delegate to memcpy(), however when MEM\_DEBUG is defined we use a generic iterator version, which allows the use of a range-checked iterator (such as [Checked-Iterator](#)) for tracking down memory-access related bugs.

Definition in file [safecopy.H](#).

### 11.355.2 Function Documentation

#### 11.355.2.1 `template<class Iter, class ConstIter> void genericcopy (Iter dest, ConstIter src, unsigned int count) [inline]`

Element-by-element copy a series forward iterators.

Definition at line 51 of file safecopy.H.

Referenced by safecopy().

#### 11.355.2.2 `template<class T> void safecopy (T * dest, const T * src, unsigned int count) [inline]`

A specialized version of [safecopy\(\)](#) for pointers.

This version uses memcpy() internally for maximum efficiency if T is a "trivial" type, otherwise it uses [genericcopy\(\)](#). **Note however** that safecopy's third parameter is number-of-elements, **NOT** number of bytes (i.e., unlike memcpy).

Definition at line 86 of file safecopy.H.

References copy().

Referenced by Hmax::origGetC2(), and Image< T >::setBorders().

#### 11.355.2.3 `template<class Iter, class ConstIter> void safecopy (Iter dest, ConstIter src, unsigned int count) [inline]`

A generic version of [safecopy\(\)](#) for iterators.

Definition at line 59 of file safecopy.H.

References genericcopy().

## 11.356 sal\_count.C File Reference

```
#include "getSaliency.H"
#include "Image.H"
#include "log.H"
#include "Raster.H"
#include "Image_DrawOps.H"
#include <stdio.h>
#include <iostream>
#include <vector>
#include "ModelManager.H"
#include "ModelOptionDefs.H"
#include "fancynorm.H"
#include "ShapeEstimator.H"
#include "ChannelBase.H"
Include dependency graph for sal_count.C:
```

### Defines

- #define [DEFAULT\\_MAX\\_TIME](#) 0.7  
*max time in seconds*

### Functions

- int [main](#) (const int argc, const char \*\*argv)  
*The main routine that reads the image, processes is and writes out the saliency info.*

### 11.356.1 Detailed Description

Print out salient locations for an image

Definition in file [sal\\_count.C](#).



## 11.356.2 Define Documentation

### 11.356.2.1 `#define DEFAULT_MAX_TIME 0.7`

max time in seconds

Usage: "sal\_count image.ppm [max\_time [traj\_file]]" This program is an example of how to use getSaliency (see [getSaliency.H](#))

Definition at line 57 of file sal\_count.C.

Referenced by main().

## 11.356.3 Function Documentation

### 11.356.3.1 `int main (const int argc, const char ** argv)`

The main routine that reads the image, processes it and writes out the saliency info.

Definition at line 63 of file sal\_count.C.

References `DEFAULT_MAX_TIME`, `FULLTRACE`, `Image< T >::getHeight()`, `Image< T >::getWidth()`, `main()`, `OPTEXP_CORE`, `Raster::ReadRGB()`, and `Raster::WriteRGB()`.

## 11.357 saliency.C File Reference

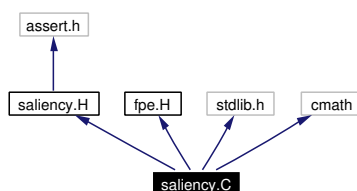
```
#include "saliency.H"
```

```
#include "fpe.H"
```

```
#include <stdlib.h>
```

```
#include <cmath>
```

Include dependency graph for saliency.C:



### Functions

- bool `isFinite` (const `byte` &arg)  
*Function to check for finite value.*
- bool `isFinite` (const `int16` &arg)  
*Function to check for finite value.*
- bool `isFinite` (const `int32` &arg)  
*Function to check for finite value.*
- bool `isFinite` (const float &arg)  
*Function to check for finite value.*
- bool `isFinite` (const double &arg)  
*Function to check for finite value.*
- int `randomUpTo` (int n)  
*Generate a random number between 0 and n (inclusive).*
- void `randShuffle` (int \*array, const int size)  
*Randomly shuffle given array.*

## Variables

- `uint initCount = 0`

### 11.357.1 Detailed Description

Functions used throughout the toolkit

Definition in file [saliency.C](#).

### 11.357.2 Function Documentation

#### 11.357.2.1 `bool isFinite (const double & arg)`

Function to check for finite value.

Definition at line 53 of file saliency.C.

Referenced by `isFinite()`.

#### 11.357.2.2 `bool isFinite (const float & arg)`

Function to check for finite value.

Definition at line 51 of file saliency.C.

#### 11.357.2.3 `bool isFinite (const int32 & arg)`

Function to check for finite value.

Definition at line 49 of file saliency.C.

#### 11.357.2.4 `bool isFinite (const int16 & arg)`

Function to check for finite value.

Definition at line 47 of file saliency.C.

#### 11.357.2.5 `bool isFinite (const byte & arg)`

Function to check for finite value.

Definition at line 45 of file saliency.C.

**11.357.2.6 int randomUpTo (int *n*)**

Generate a random number between 0 and *n* (inclusive).

Definition at line 56 of file saliency.C.

Referenced by ImageSpring< T >::getStats(), ImageSpring< T >::getStatsDist(), SimulationViewerCompress::getTraj(), and main().

**11.357.2.7 void randShuffle (int \* *array*, const int *size*)**

Randomly shuffle given array.

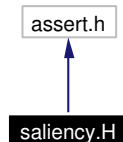
Definition at line 62 of file saliency.C.

Referenced by main().

## 11.358 saliency.H File Reference

```
#include <assert.h>
```

Include dependency graph for saliency.H:



This graph shows which files directly or indirectly include this file:

### Classes

- struct [type\\_if](#)  
*Makes a type a typedef for T only if B is true.*
- struct **type\_if**< T, true >
- struct [type\\_with\\_N\\_bits](#)  
*A compile-time check that T has N bits.*
- class [ToolkitInit](#)  
*A helper class for program-wide initialization.*

### Defines

- #define [ASSERT](#)(x) `assert(x)`  
*to have some low-level checking, set [ASSERT\(\)](#) to `assert()`:*

### Typedefs

- typedef [type\\_with\\_N\\_bits](#)< unsigned char, 8 >::type [byte](#)  
*8-bit unsigned integer*
- typedef [type\\_with\\_N\\_bits](#)< short, 16 >::type [int16](#)  
*16-bit signed integer*

- typedef [type\\_with\\_N\\_bits](#)< unsigned short, 16 >::type [uint16](#)  
*16-bit unsigned integer*
- typedef [type\\_with\\_N\\_bits](#)< int, 32 >::type [int32](#)  
*32-bit signed integer*
- typedef [type\\_with\\_N\\_bits](#)< unsigned int, 32 >::type [uint32](#)  
*32-bit unsigned integer*
- typedef [type\\_with\\_N\\_bits](#)< long long int, 64 >::type [int64](#)  
*64-bit signed integer*
- typedef [type\\_with\\_N\\_bits](#)< unsigned long long int, 64 >::type [uint64](#)  
*64-bit unsigned integer*
- typedef unsigned int [uint](#)  
*Canonical unsigned int.*

## Functions

- template<class T> bool [ODD](#) (const T &x)  
*True if x is odd.*
- template<class T> T [ABS](#) (const T &x)  
*Returns the absolute value of x.*
- template<class T> int [SGN](#) (const T &x)  
*Returns -1 if x is negative, or 1 if x is positive.*
- template<class T> T [SQR](#) (const T &x)  
*Returns the square of x.*
- template<class T, class T2> T [CLAMP](#) (const T &val, const T2 &bound1, const T2 &bound2)  
*Returns the number closest to val and between bound1 and bound2.*
- bool [isFinite](#) (const [byte](#) &arg)  
*Function to check for finite value.*
- bool [isFinite](#) (const [int16](#) &arg)

*Function to check for finite value.*

- bool [isFinite](#) (const [int32](#) &arg)

*Function to check for finite value.*

- bool [isFinite](#) (const float &arg)

*Function to check for finite value.*

- bool [isFinite](#) (const double &arg)

*Function to check for finite value.*

- int [randomUpTo](#) (int n)

*Generate a random number between 0 and n (inclusive).*

- void [randShuffle](#) (int \*array, const int size)

*Randomly shuffle given array.*

## Variables

- [ToolkitInit](#) [toolkitInit\\_\\_](#) (true)

### 11.358.1 Detailed Description

Type definitions and macros

Definition in file [saliency.H](#).

### 11.358.2 Define Documentation

#### 11.358.2.1 `#define ASSERT(x) assert(x)`

to have some low-level checking, set [ASSERT\(\)](#) to `assert()`:

Definition at line 53 of file [saliency.H](#).

Referenced by [absDiff\(\)](#), [OptionSet::add\(\)](#), [OptionParser::addAlias\(\)](#), [Image< T >::addBGnoise\(\)](#), [Image< T >::addBGnoise2\(\)](#), [Image< T >::addScaled\(\)](#), [Image< T >::addWeighted\(\)](#), [Vector2D::angle\(\)](#), [DirectFeedChannel::applyMaxNorm\(\)](#), [VisualEvent::assign\(\)](#), [Image< T >::attenuateBorders\(\)](#), [average\(\)](#), [averageWeighted\(\)](#), [ImageSet< T >::back\(\)](#), [ImageCache< T >::back\(\)](#), [BeobotSensation::BeobotSensation\(\)](#), [SingleChannel::biasFeatures\(\)](#), [binomialKernel\(\)](#),

Rectangle::bottom(), BeoChip::calibrateAnalog(), BeoChip::calibratePulse(), BeoChip::calibrateServo(), BeoChip::captureAnalog(), BeoChip::capturePulse(), SingleChannel::centerSurround(), centerSurround(), DummyChannel::centerSurround(), centerSurroundDiff(), centerSurroundDiffSingleOpponent(), centerSurroundSingleOpponent(), chamfer34(), SingleChannelBeoServer::check(), clampedDiff(), colorStain(), SurpriseModelSP::combineFrom(), SurpriseModelSG::combineFrom(), MultiDirectFeedChannel::combineOutputs(), ShapeEstimator::compute(), computeCMAP(), computeCMAP2(), DirectFeedChannel::computeOutput(), TCPcommunicator::contact(), contour2D(), contour2D8(), ContourConnection::ContourConnection(), contrastModulate(), convolveHmax(), copy(), corrcoeff(), corrpach(), Image< T >::countThresh(), LevelSpec::csToIndex(), dct(), OptionParser::destroy(), dilateImg(), Dims::Dims(), MultiDirectFeedChannel::directFeed(), StraightLine2D::direction(), distance(), Vector2D::distance(), doAddWeighted(), SoxChannel::doInput(), SingleChannel::doInput(), MultiDirectFeedChannel::doInput(), HueChannel::doInput(), PN03contrastChannel::doInput(), InformationChannel::doInput(), ScorrChannel::doInput(), TcorrChannel::doInput(), VarianceChannel::doInput(), EntropyChannel::doInput(), EndPointChannel::doInput(), XJunctionChannel::doInput(), TJunctionChannel::doInput(), LJunctionChannel::doInput(), JunctionChannel::doInput(), MotionChannel::doInput(), MultiColorBandChannel::doInput(), OrientationChannel::doInput(), SOColorChannel::doInput(), ColorChannel::doInput(), MultiConvolveChannel::doInput(), FlickerChannel::doInput(), RGBConvolveChannel::doInput(), OrientationChannel::doInteractions(), Vector2D::dotProduct(), doubleOpp(), ImageCacheAvg< T >::doWhenAdd(), ImageCacheAvg< T >::doWhenRemove(), downSize(), downSizeClean(), BitObject::drawBoundingBox(), Image< T >::drawCircle(), Image< T >::drawCross(), Image< T >::drawDisk(), Image< T >::drawDiskCheckTarget(), Image< T >::drawGrid(), XWindow::drawImage(), Image< T >::drawLine(), BitObject::drawOutline(), Image< T >::drawPatch(), Image< T >::drawRect(), BitObject::drawShape(), contourRun::dumpEnergySigmoid(), Image< T >::embed(), erodeImg(), SaliencyMapFast::evolve(), SaliencyMapStd::evolve(), featurePoolHmax(), Image< T >::findMax(), Image< T >::findMin(), findMontyMap(), Rectangle::fitToDims(), Image< T >::flood(), floodClean(), floodCleanBB(), ImageSet< T >::front(), ImageCache< T >::front(), BeobotMemory::generatePresentAction(), GACHromosome::get\_gene(), BeoChip::getAnalog(), BeoChip::getAnalogRaw(), BeoSub::getAveragedDepth(), BeoSub::getAveragedHeading(), BeobotSensation::getCentroid(), ShapeEstimator::getChannel(), DirectFeedChannel::getCoeff(), getComponents(), KalmanFilter::getCost(), KalmanFilter::getCovariances(), KalmanFilter::getEstimate(), VisualCortexEyeMvt::getEyePos(), VisualCortex::getFeatures(), VisualEventSet::getFOE(), ImageSet< T >::getImage(), ImageCache< T >::getImage(), ImageSet< T >::getImageMut(), JetSpec::getIndexBase(), BeobotVisualCortex::getInputSize(), ShapeEstimator::getIORMask(), readConfig::getItemValueB(), readConfig::getItemValueC(), readConfig::getItemValueF(), readConfig::getItemValueS(), MbariResultViewer::getLabel(), Image< T >::getMean(), Image< T >::getMinMax(), Image< T >::getMinMaxAvg(), getMinMaxC(), JetSpec::getNbIndexRanges(), ImageSpring< T >::getNeighbor(), ShapeEstimator::getObjectArea(), ShapeEstimator::getObjectMask(), BitObject::getObjectMask(), DirectFeedChannel::getOutput(), ImageSet< T >::getPixel(), Image-



Set< T >::getPixelNI(), Vector2D::getPoint2D(), BeoChip::getPulse(), BeoChip::getPulseRaw(), ShapeEstimator::getPyramid(), BeobotSensation::getQuickLayout(), SingleChannel::getRawCSmap(), getRGBY(), getRGBYsimple(), Image< T >::getSample(), MbariResultViewer::getSaveEventClipNum(), BeoChip::getServo(), BeoChip::getServoRaw(), ShapeEstimator::getSingleChannel(), ShapeEstimator::getSmoothMask(), KalmanFilter::getSpeed(), KalmanFilter::getStateVector(), ImageSpring< T >::getStats(), ImageSpring< T >::getStatsDist(), Image< T >::getStdev(), DirectFeedChannel::getSubmap(), SingleChannel::getSubmapName(), DirectFeedChannel::getSubmapName(), ShapeEstimator::getSubmapNum(), SurpriseMap< T >::getSurpriseImage(), VisualEvent::getToken(), RadioDecoder::getVal(), Image< T >::getVal(), Image< T >::getValInterp(), BeoSubMotor::getValue(), Image< T >::getVector(), Image< T >::getVectorColumn(), Image< T >::getVectorRow(), ShapeEstimator::getWinner(), ShapeEstimator::getWinningLabel(), ShapeEstimator::getWinningMap(), getYIQ(), getYUV(), goodness\_map(), IEEE1394grabber::grab(), BeoSub::grab(), AudioGrabber::grab(), Rectangle::height(), highThresh(), ImageSpring< T >::ImageSpring(), LevelSpec::indexToCS(), infoMeasure(), SurpriseImage< T >::init(), LevelSpec::init(), Jet< T >::init(), VisualEventSet::initiateEvents(), MultiDirectFeedChannel::inputPyramid(), DirectFeedChannel::inputPyramid(), MultiDirectFeedChannel::inputPyramidVector(), StraightLine2D::intersect(), Vector2D::isCollinear(), Image< T >::isFinite(), StraightLine2D::isIdentical(), StraightLine2D::isOrthogonal(), Vector2D::isOrthogonal(), StraightLine2D::isParallel(), StraightLine2D::isPointOnLine(), Vector2D::isZero(), junctionFilterFull(), junctionFilterPartial(), learningCoeff(), Rectangle::left(), LevelSpec::LevelSpec(), Image< T >::lowThresh(), Image< T >::lowThreshAbs(), luminance(), Image< T >::luminanceNormalize(), luminanceNormalize(), main(), makeRGB(), makeSparseMap(), BeobotVisualCortex::masterCollect(), BeobotVisualCortex::masterProcess(), matrixMult(), maxNormalizeFancy(), maxNormalizeFancyFast(), maxNormalizeFancyLandmark(), maxNormalizeFancyWeak(), maxNormalizeLandmark(), maxNormalizeStd(), mean(), CameraControl::moveCamXYFrame(), SurpriseImage< T >::neighborhoods(), Image< T >::normalize(), Vector2D::normalize(), normalizeC(), Image< T >::operator\*(), operator\*=(), Image< T >::operator\*=(), Image< T >::operator+(), operator+=(), Image< T >::operator+=(), Image< T >::operator-(), operator-=(), Image< T >::operator-=(), Image< T >::operator/(), Vector2D::operator/(), operator/=(), Image< T >::operator/=(), Vector2D::operator/=(), Jet< T >::operator=(), BeobotSensation::operator=(), operator>>(), Image< T >::operator[](), optConvolve(), overlay(), overlayStain(), CameraControl::panAbsolute(), CameraControl::panRelative(), BeobotMemory::passedSensation(), Image< T >::paste(), Image< T >::pasteGabor(), PngParser::PngParser(), StraightLine2D::point(), ImageSet< T >::pop\_back(), ImageSet< T >::pop\_front(), ImageCache< T >::pop\_front(), BeobotMemory::presentActionRecommendation(), BeobotMemory::presentSensation(), pSNRcolor(), BeoSubMotor::pulseValue(), SingleChannel::pyramid(), SingleChannel::pyramidTime(), SingleChannel::pyrMut(), ParamMap::queryStringParam(), raodistance(), readConfig::readFileValueC(), readConfig::readFileValueF(), readConfig::readFileValueI(), readConfig::readFileValueName(), readConfig::readFileValueNameC(), readConfig::readFileValueNameF(), readConfig::readFileValueS(), Beowulf::receive(),

receiveCMAPS(), TCPcommunicator::receiveIt(), Image< T >::rectify(), Image< T >::replaceVal(), BitObject::reset(), Rectangle::right(), RMSerr(), MbariResultViewer::saveSingleEventFrame(), segmentLandmark(), segmentObject(), segmentObjectClean(), TCPcommunicator::send(), Beowulf::send(), GACHromosome::set\_breedings(), GAPopulation::set\_chromosome(), GACHromosome::set\_fitness(), GACHromosome::set\_gene(), GACHromosome::set\_linear\_fitness(), Image< T >::setBorders(), DirectFeedChannel::setCoeff(), BeoChip::setDigitalOut(), readConfig::setFileValue(), readConfig::setItemValue(), contourRun::setIterations(), BitObject::setMaxMinAvgIntensity(), SingleChannel::setPyramid(), BeoChip::setServo(), BeoChip::setServoRaw(), Image< T >::setVal(), Image< T >::setValMask(), BeoSubMotor::setValue(), setYUV(), Image< T >::sigmoid(), SingleChannel::singleOpponentInput(), BeobotVisualCortex::slaveProcess(), Image< T >::speckleNoise(), Image< T >::splitPosNeg(), Image< T >::square(), SingleChannel::storePyramid(), SingleChannel::storePyramid2(), ComplexChannel::subChanFromSubmapNum(), sum(), SurpriseImage< T >::surprise(), takeMax(), takeMin(), TCPcommunicator::terminateCli(), CameraControl::tiltAbsolute(), CameraControl::tiltRelative(), Rectangle::top(), toRGB(), KalmanFilter::update(), VisualEventSet::updateEvents(), GACHromosome::use\_breeding(), Image< T >::varianceNormalize(), SingleChannelBeo::waitForOutput(), warp3D(), Rectangle::width(), Image< T >::wilsonNormalize(), Vector2D::x(), Vector2D::y(), and ToolkitInit::~ToolkitInit().

### 11.358.3 Typedef Documentation

#### 11.358.3.1 typedef [type\\_with\\_N\\_bits](#)<unsigned char, 8>::type byte

8-bit unsigned integer

Definition at line 85 of file saliency.H.

Referenced by SSC::calibToRaw(), BitObject::doesIntersect(), EntropyChannel::doInput(), BitObject::drawOutline(), extractBitObjects(), fromRGB555(), fromRGB555bs(), fromRGB565(), fromRGB565bs(), BitObject::getObjectMask(), SimulationViewerCompress::getTraj(), BeoChip::lcdLoadFont(), BeoChip::lcdSendRaw(), BeoChip::lcdSetAnimation(), VisualCortex::learn(), learningCoeff(), main(), SSC::rawToCalib(), readBinImage(), BitObject::reset(), BitObject::setMaxMinAvgIntensity(), showAllObjects(), BeobotVisualCortex::singleCPUprocess(), and toYUV422().

#### 11.358.3.2 typedef [type\\_with\\_N\\_bits](#)<short, 16>::type int16

16-bit signed integer

Definition at line 88 of file saliency.H.

Referenced by isFinite().

**11.358.3.3 typedef [type\\_with\\_N\\_bits](#)<int, 32>::type int32**

32-bit signed integer

Definition at line 94 of file saliency.H.

**11.358.3.4 typedef [type\\_with\\_N\\_bits](#)<long long int, 64>::type int64**

64-bit signed integer

Definition at line 100 of file saliency.H.

Referenced by `Timer::get()`, `Timer::getReset()`, and `Timer::Timer()`.

**11.358.3.5 typedef unsigned int [uint](#)**

Canonical unsigned int.

Definition at line 106 of file saliency.H.

Referenced by `SingleChannel::absSumCoeffs()`, `DirectFeedChannel::absSumCoeffs()`, `ComplexChannel::absSumCoeffs()`, `SortedLimitedList< T >::add()`, `VisualCortex::addChannel()`, `TCPmessage::addImageSet()`, `VisualObjectDB::addObject()`, `ComplexChannel::addSubChan()`, `ModelComponent::addSubComponent()`, `Histogram::addValue()`, `MultiColorBandChannelSurprise::band()`, `MultiColorBandChannel::band()`, `BeoChip::BeoChip()`, `VisualCortex::biasFeatures()`, `SingleChannel::biasFeatures()`, `ComplexChannel::biasFeatures()`, `buildPyrGabor()`, `SoxChannel::buildSubChans()`, `EndPointChannelSurprise::buildSubChans()`, `TJunctionChannelSurprise::buildSubChans()`, `LJunctionChannelSurprise::buildSubChans()`, `MotionChannelSurprise::buildSubChans()`, `MultiColorBandChannelSurprise::buildSubChans()`, `OrientationChannelSurprise::buildSubChans()`, `MotionChannelBeo::buildSubChans()`, `MultiColorBandChannelBeo::buildSubChans()`, `OrientationChannelBeo::buildSubChans()`, `StereoChannel::buildSubChans()`, `EndPointChannel::buildSubChans()`, `TJunctionChannel::buildSubChans()`, `LJunctionChannel::buildSubChans()`, `MotionChannel::buildSubChans()`, `MultiColorBandChannel::buildSubChans()`, `OrientationChannel::buildSubChans()`, `cellArr2ImageSetVec()`, `SingleChannel::centerSurround()`, `centerSurround()`, `DummyChannel::centerSurround()`, `centerSurroundDiff()`, `centerSurroundDiffSingleOpponent()`, `centerSurroundSingleOpponent()`, `VisualCortex::channel()`, `CompositeColorChannel::channel()`, `VisualCortex::channelPtr()`, `SingleChannelBeoServer::check()`, `SingleChannel::clampCoeffs()`, `DirectFeedChannel::clampCoeffs()`, `ComplexChannel::clampCoeffs()`, `clampedDiff()`, `VisualCortex::clampParams()`, `SingleChannel::cleanCoeffs()`, `DirectFeedChannel::cleanCoeffs()`, `ComplexChannel::cleanCoeffs()`, `VisualCortex::cleanParams()`, `VisualEventSet::cleanUp()`, `LevelSpec::clevOK()`, `ColorBandChannel::ColorBandChannel()`, `ColorBandChannelSurprise::ColorBandChannelSurprise()`, `SoxChannel::combineOutputs()`, `MultiDirectFeedChannel::combineOutputs()`, `ComplexChannel::combineOutputs()`,

SingleChannel::combineSubMaps(), ObjectBase::compareAll(), CompositeColorChannel::CompositeColorChannel(), CompositeColorChannelSurprise::CompositeColorChannelSurprise(), ShapeEstimator::compute(), FrameSeries::computeFileName(), DirectFeedChannel::computeOutput(), SingleChannel::computeSubmap(), DummyChannelSurprise::computeSubmap(), JunctionChannelSurprise::computeSubmap(), PedestrianChannelSurprise::computeSubmap(), TemplateMatchChannelSurprise::computeSubmap(), DirectionChannelSurprise::computeSubmap(), ColorBandChannelSurprise::computeSubmap(), FlickerChannelSurprise::computeSubmap(), GaborChannelSurprise::computeSubmap(), YellowChannelSurprise::computeSubmap(), BlueChannelSurprise::computeSubmap(), GreenChannelSurprise::computeSubmap(), RedChannelSurprise::computeSubmap(), SOYellowBlueChannelSurprise::computeSubmap(), SOBlueYellowChannelSurprise::computeSubmap(), SOGreenRedChannelSurprise::computeSubmap(), SORedGreenChannelSurprise::computeSubmap(), BlueYellowChannelSurprise::computeSubmap(), RedGreenChannelSurprise::computeSubmap(), ConvolveChannelSurprise::computeSubmap(), IntensityChannelSurprise::computeSubmap(), KDTree::createKDTree(), LevelSpec::csOK(), SingleChannel::csToIndex(), LevelSpec::csToIndex(), OptionSet::debugDump(), LevelSpec::delMax(), LevelSpec::delMin(), LevelSpec::delOK(), MotionChannelSurprise::dirChan(), MotionChannel::dirChan(), MultiDirectFeedChannel::directFeed(), DirectionChannel::DirectionChannel(), DirectionChannelSurprise::DirectionChannelSurprise(), DisparityChannel::DisparityChannel(), StereoChannel::dispChan(), MbariResultViewer::display(), PsychoDisplay::displayEyeTrackerCalibration(), doAddWeighted(), doApplyBiases(), doClear(), doEnergyNorm(), VisualEventSet::doesEventExist(), SoxChannel::doInput(), MultiDirectFeedChannel::doInput(), PN03contrastChannel::doInput(), InformationChannel::doInput(), ScorrChannel::doInput(), VarianceChannel::doInput(), EntropyChannel::doInput(), StereoChannel::doInput(), EndPointChannel::doInput(), XJunctionChannel::doInput(), TJunctionChannel::doInput(), LJunctionChannel::doInput(), JunctionChannel::doInput(), MotionChannel::doInput(), OrientationChannel::doInput(), CompositeColorChannel::doInput(), DisparityChannel::doInput(), OrientationChannel::doInteractions(), doLowThresh(), doLowThreshAbs(), doOneNormalize(), doRectify(), doSqrt(), VisualCortex::drawResults(), ComplexChannel::drawResults(), VisualEventSet::drawTokens(), VisualCortexEyeMvt::evolve(), Brain::evolve(), SimulationViewerEyeMvt::eyeMovement(), VisualCortex::fillJet(), ComplexChannel::fillJet(), VisualCortex::fillJetNI(), ComplexChannel::fillJetNI(), ComplexChannel::fillJetSpec(), VisualObjectDB::filter(), TemplateMatchChannel::findBestMatch(), VisualCortex::findLikelihood(), SingleChannel::findLikelihood(), ComplexChannel::findLikelihood(), Histogram::findMax(), VisualCortex::findWeight(), SingleChannel::findWeight(), ComplexChannel::findWeight(), MbariResultViewer::freeMem(), OrientationChannelSurprise::gabor(), OrientationChannel::gabor(), GaborChannel::GaborChannel(), GaborChannelSurprise::GaborChannelSurprise(), VisualEventSet::getAllClosedFrameNum(), MbariResultViewer::getAvgCacheSize(), getCentroidFirstLast(), SingleChannel::getCoeff(), DirectFeedChannel::getCoeff(), ImageSet< T >::getDepth(), VisualEvent::getEndFrame(), VisualEventSet::getEventByNumber(), VisualEvent::getEventNum(), VisualEventSet::getEventsForFrame(),

ModelManager::getExtraArg(), MexModelManager::getExtraMexArg(), VisualCortexEyeMvt::getEyePos(), VisualObject::getFeatureDistSq(), VisualCortex::getFeatures(), SingleChannel::getImage(), ImageSet< T >::getImage(), ImageCache< T >::getImage(), ImageSet< T >::getImageMut(), VisualCortex::getJetSpec(), MbariResultViewer::getLabel(), ImageCache< T >::getMaxSize(), SingleChannel::getMean(), SoxChannel::getNonlinearResponse(), VisualEvent::getNumberOfFrames(), MbariResultViewer::getNumFromStrings(), VisualObjectDB::getObject(), ShapeEstimator::getObjectMask(), VisualCortexEyeMvt::getOutput(), VisualCortex::getOutput(), TcorrChannel::getOutput(), ImageSet< T >::getPixel(), ImageSet< T >::getPixelNI(), VisualEvent::getPropertyVector(), PropertyVectorSet::getPropertyVectorForEvent(), SingleChannel::getRawCSmap(), MbariResultViewer::getSaveEventClipNum(), SingleChannel::getSigma(), ShapeEstimator::getSingleChannel(), VisualEvent::getStartFrame(), SingleChannel::getSubmap(), DirectFeedChannel::getSubmap(), ComplexChannel::getSubmap(), HueChannel::getSubmap(), SingleChannel::getSubmapName(), DirectFeedChannel::getSubmapName(), ComplexChannel::getSubmapName(), HueChannel::getSubmapName(), SurpriseMap< T >::getSurpriseImage(), SingleChannelSurprise::getSurpriseMap(), VisualEvent::getToken(), VisualEventSet::getTokens(), SimulationViewerCompress::getTraj(), FeatureVector::getValue(), Histogram::getValue(), ImageCache< T >::ImageCache(), ImageCacheAvg< T >::ImageCacheAvg(), ImageCacheMinMax< T >::ImageCacheMinMax(), ImageSet< T >::ImageSet(), SortedLimitedList< T >::indexOf(), LevelSpec::indexOK(), LevelSpec::indexToCS(), SurpriseMap< T >::init(), LevelSpec::init(), KalmanFilter::init(), MPEGStream::initEncode(), DirectFeedChannel::initializeCoeffs(), VisualCortexEyeMvt::input(), VisualCortex::input(), MultiDirectFeedChannel::inputPyramid(), MultiDirectFeedChannel::inputPyramidVector(), isDyadic(), VisualEvent::isFrameOk(), isHomogeneous(), ImageSet< T >::isEmpty(), SingleChannel::killCaches(), VisualCortex::learn(), SingleChannel::learn(), ComplexChannel::learn(), LevelSpec::LevelSpec(), LevelSpec::levMax(), LevelSpec::levMin(), main(), LevelSpec::mapLevel(), VisualObject::match(), VisualObjectDB::matchObject(), LevelSpec::maxDepth(), LevelSpec::maxIndex(), mexArr2Vector(), MultiDirectFeedChannel::MultiDirectFeedChannel(), FeatureVector::normalize(), SingleChannel::normalizeCoeffs(), DirectFeedChannel::normalizeCoeffs(), ComplexChannel::normalizeCoeffs(), DisparityChannel::normalizeDispMap(), VisualCortex::normalizeParams(), VisualCortex::numChannels(), SingleChannel::numChans(), DirectFeedChannel::numChans(), ComplexChannel::numChans(), VisualEventSet::numEvents(), ModelManager::numExtraArgs(), MexModelManager::numExtraMexArgs(), VisualCortexEyeMvt::numHumans(), MbariResultViewer::numSaveEventClips(), SoxChannel::numScales(), ModelComponent::numSubComp(), SingleChannel::numSubmaps(), DirectFeedChannel::numSubmaps(), ComplexChannel::numSubmaps(), HueChannel::numSubmaps(), operator \*(), operator +(), operator -(), operator /(), ImageSet< T >::operator[](), ImageCache< T >::operator[](), orientedFilterSet(), MbariResultViewer::output(), ComplexChannel::outputAvailable(), MbariResultViewer::outputResultFrame(), ModelManager::parseCommandLine(), MexModelManager::parseCommandLine(), MbariResultViewer::parseSaveEventNums(), Point2DVec2mexArr(), SingleChannel::postProcessMap(),

CloseButtonListener::pressedAnyCloseButton(), ModelManager::printHelp(), SingleChannel::pyramid(), SingleChannel::pyramidTime(), SingleChannel::pyrMut(), RadioDecoder::RadioDecoder(), rangeOf(), readBinImage(), VisualCortex::readFrom(), SingleChannel::readFrom(), ComplexChannel::readFrom(), PropertyVectorSet::readFromStream(), VisualCortex::reco(), SingleChannel::reco(), ComplexChannel::reco(), reduce(), ModelComponent::removeSubComponent(), TestSuite::requireEq(), rescale(), SurpriseMap< T >::reset(), SingleChannel::reset(), ImageSet< T >::reset(), VisualObjectDB::save(), VisualObject::save(), MbariResultViewer::save(), VisualCortex::saveResults(), SingleChannel::saveResults(), ComplexChannel::saveResults(), SingleChannel::setCoeff(), DirectFeedChannel::setCoeff(), ComplexChannel::setListener(), SortedLimitedList< T >::setMaxElements(), ImageCache< T >::setMaxSize(), SingleChannel::setMean(), VisualCortexConfigurator::setModelParamString(), MultiDirectFeedChannel::setNumChans(), SingleChannel::setSigma(), ImageSet< T >::size(), ImageCache< T >::size(), LevelSpec::slevOK(), Histogram::smooth(), SSC::SSC(), VisualCortexEyeMvt::start1(), SimulationViewerCompress::start1(), ComplexChannel::start2(), SoxChannel::subChan(), ComplexChannel::subChan(), ComplexChannel::subChanFromSubmapNum(), ModelComponent::subComponent(), sum(), SurpriseMap< T >::surprise(), takeSlice(), FeatureVector::threshold(), Token::Token(), Vector2mexArr(), VisualEventSet::VisualEventSet(), SingleChannelBeo::waitForOutput(), writeBinImage(), VisualEvent::writePositions(), VisualCortex::writeTo(), SingleChannel::writeTo(), ComplexChannel::writeTo(), VisualEventSet::writeToStream(), VisualEvent::writeToStream(), and PropertyVectorSet::writeToStream().

### 11.358.3.6 `typedef type_with_N_bits<unsigned short, 16>::type uint16`

16-bit unsigned integer

Definition at line 91 of file saliency.H.

### 11.358.3.7 `typedef type_with_N_bits<unsigned int, 32>::type uint32`

32-bit unsigned integer

Definition at line 97 of file saliency.H.

### 11.358.3.8 `typedef type_with_N_bits<unsigned long long int, 64>::type uint64`

64-bit unsigned integer

Definition at line 103 of file saliency.H.

Referenced by Brain::evolve(), Timer::get(), Timer::getReset(), main(), CINNIC::RunSimpleImage(), and PsychoDisplay::waitVsync().

## 11.358.4 Function Documentation

### 11.358.4.1 `template<class T> T ABS (const T & x) [inline, static]`

Returns the absolute value of  $x$ .

Definition at line 119 of file saliency.H.

Referenced by `BeoSub::advance()`, and `Image< T >::drawLine()`.

### 11.358.4.2 `template<class T, class T2> T CLAMP (const T & val, const T2 & bound1, const T2 & bound2) [inline, static]`

Returns the number closest to *val* and between *bound1* and *bound2*.

The bounds may be specified in either order (i.e., either (lo, hi) or (hi, lo)).

Definition at line 133 of file saliency.H.

Referenced by `SingleChannel::clampCoeffs()`, `DirectFeedChannel::clampCoeffs()`, `ChannelBase::clampWeight()`, `Image< T >::drawCross()`, `Image< T >::drawPatch()`, and `warp3D()`.

### 11.358.4.3 `bool isFinite (const double & arg)`

Function to check for finite value.

Definition at line 53 of file saliency.C.

Referenced by `isFinite()`.

### 11.358.4.4 `bool isFinite (const float & arg)`

Function to check for finite value.

Definition at line 51 of file saliency.C.

### 11.358.4.5 `bool isFinite (const int32 & arg)`

Function to check for finite value.

Definition at line 49 of file saliency.C.

### 11.358.4.6 `bool isFinite (const int16 & arg)`

Function to check for finite value.

Definition at line 47 of file saliency.C.

References int16.

#### **11.358.4.7    bool isFinite (const [byte](#) & *arg*)**

Function to check for finite value.

Definition at line 45 of file saliency.C.

#### **11.358.4.8    template<class T> bool ODD (const T & *x*)   [inline, static]**

True if *x* is odd.

Definition at line 115 of file saliency.H.

#### **11.358.4.9    int randomUpTo (int *n*)**

Generate a random number between 0 and *n* (inclusive).

Definition at line 56 of file saliency.C.

Referenced by ImageSpring< T >::getStats(), ImageSpring< T >::getStatsDist(), SimulationViewerCompress::getTraj(), and main().

#### **11.358.4.10   void randShuffle (int \* *array*, const int *size*)**

Randomly shuffle given array.

Definition at line 62 of file saliency.C.

Referenced by main().

#### **11.358.4.11   template<class T> int SGN (const T & *x*)   [inline, static]**

Returns -1 if *x* is negative, or 1 if *x* is positive.

Definition at line 123 of file saliency.H.

Referenced by Image< T >::drawLine().

#### **11.358.4.12   template<class T> T SQR (const T & *x*)   [inline, static]**

Returns the square of *x*.

Definition at line 127 of file saliency.H.



Referenced by `avgOrient()`, `corrcoef()`, `ScorrChannel::doInput()`, `Image< T >::drawCircle()`, `Image< T >::drawDisk()`, `Image< T >::drawDiskCheckTarget()`, `Edge::find_error()`, `ImageSpring< T >::getClusteredImage()`, `ImageSpring< T >::getDistanceMasses()`, `SoxChannel::getNonlinearResponse()`, `ImageSpring< T >::getStatsDist()`, and `SaliencyMap::w()`.

## 11.359 SaliencyMap.C File Reference

```
#include "saliency.H"
#include "Image.H"
#include "SaliencyMap.H"
#include "log.H"
#include "ModelOptionDefs.H"
#include "ModelManager.H"
#include "FrameSeries.H"
#include <cmath>
```

Include dependency graph for SaliencyMap.C:

### 11.359.1 Detailed Description

Implementation for saliency map class

Definition in file [SaliencyMap.C](#).

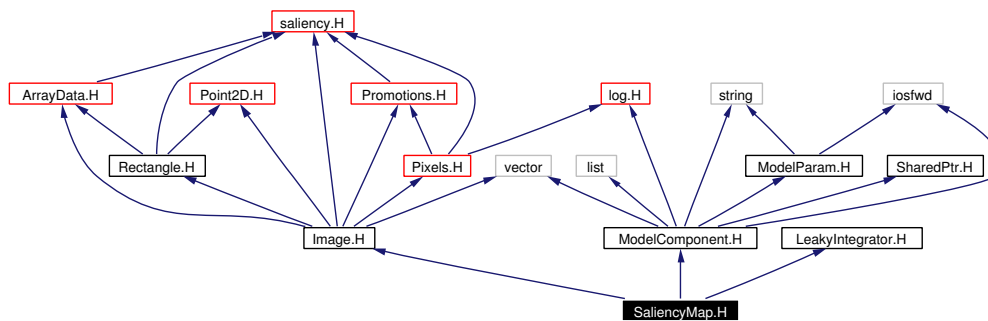
## 11.360 SaliencyMap.H File Reference

```
#include "Image.H"
```

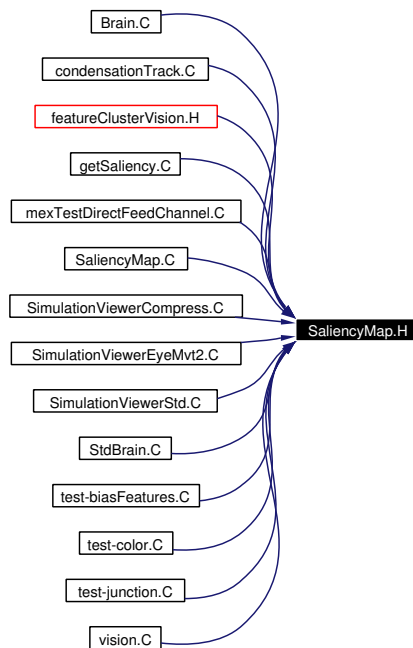
```
#include "LeakyIntegrator.H"
```

```
#include "ModelComponent.H"
```

Include dependency graph for SaliencyMap.H:



This graph shows which files directly or indirectly include this file:



## Classes

- class [SaliencyMap](#)  
*The saliency map base class.*
- class [SaliencyMapConfigurator](#)  
*[SaliencyMap](#) configurator.*
- class [SaliencyMapStd](#)  
*The standard saliency map: a 2D array of [LeakyIntegrator](#) neurons.*
- class [SaliencyMapTrivial](#)  
*The trivial saliency map: a 2D array of floats.*
- class [SaliencyMapFast](#)  
*The fast saliency map: a 2D array of floats.*

### 11.360.1 Detailed Description

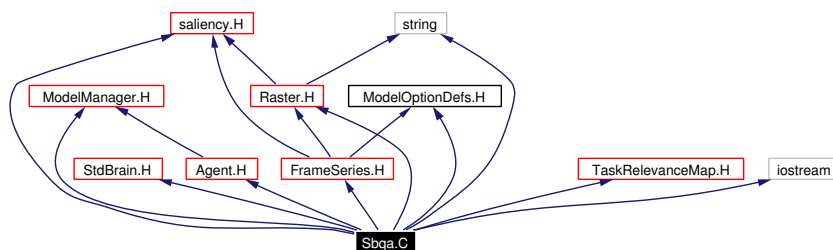
Class declarations for saliency map class

Definition in file [SaliencyMap.H](#).

## 11.361 Sbqa.C File Reference

```
#include "saliency.H"
#include "ModelManager.H"
#include "StdBrain.H"
#include "FrameSeries.H"
#include "Agent.H"
#include "ModelOptionDefs.H"
#include "Raster.H"
#include "TaskRelevanceMap.H"
#include <string>
#include <iostream>
```

Include dependency graph for Sbqa.C:



## Functions

- int **main** (const int argc, const char \*\*argv)

### 11.361.1 Detailed Description

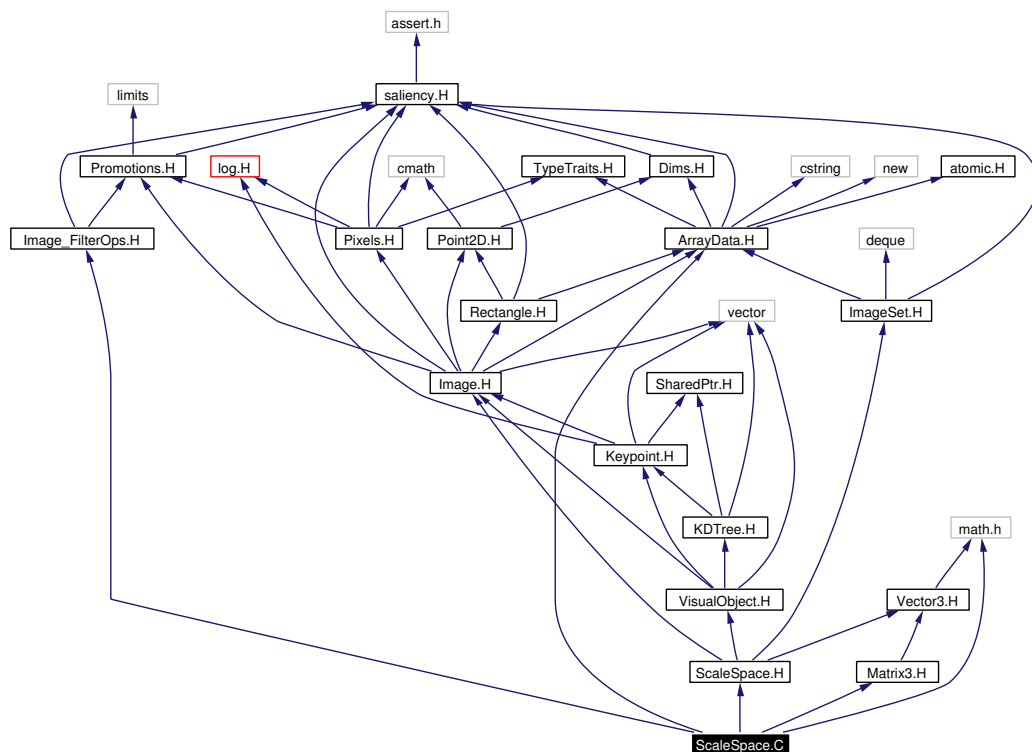
– scene based question answering model

Definition in file [Sbqa.C](#).

## 11.362 ScaleSpace.C File Reference

```
#include "ScaleSpace.H"
#include "ArrayData.H"
#include "Image_FilterOps.H"
#include "Matrix3.H"
#include <math.h>
```

Include dependency graph for ScaleSpace.C:



## Functions

- void **gradientOrientationImages** (const [Image](#)< float > &in, [Image](#)< float > &gradient, [Image](#)< float > &orientation)

### 11.362.1 Detailed Description

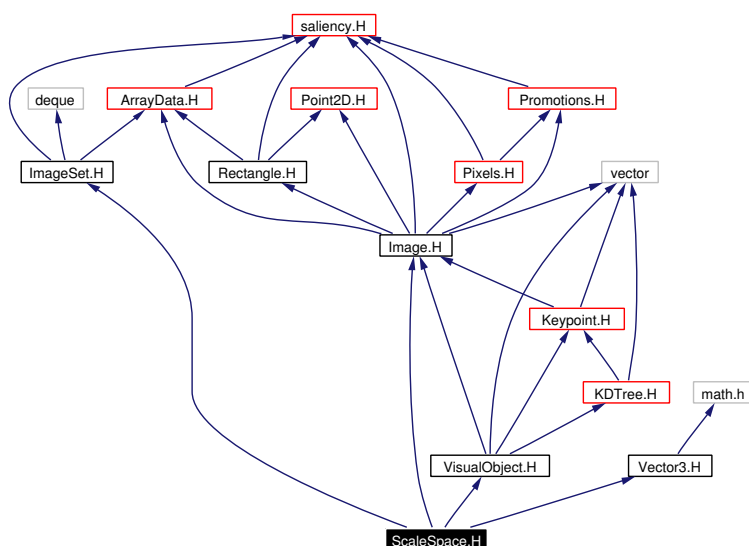
Keypoint computation for SIFT obj recognition

Definition in file [ScaleSpace.C](#).

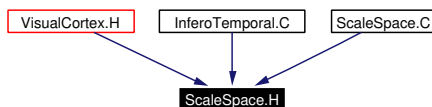
## 11.363 ScaleSpace.H File Reference

```
#include "ImageSet.H"
#include "Image.H"
#include "VisualObject.H"
#include "Vector3.H"
```

Include dependency graph for ScaleSpace.H:



This graph shows which files directly or indirectly include this file:



## Classes

- class [ScaleSpace](#)  
*ScaleSpace.*



## Variables

- const int **SCALES** = 3
- const float **SIGMA** = 1.6f
- const int **EXT** = 10
- const int **R\_EDGE** = 10
- const float **PEAK\_THRESH** = 0.001f
- const float **PI** = M\_PI

### 11.363.1 Detailed Description

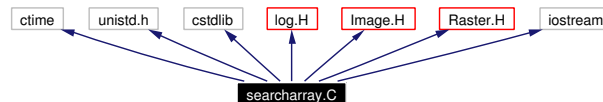
Keypoint computation for SIFT obj recognition

Definition in file [ScaleSpace.H](#).

## 11.364 searcharray.C File Reference

```
#include <ctime>
#include <unistd.h>
#include <cstdlib>
#include "log.H"
#include "Image.H"
#include "Raster.H"
#include <iostream>
```

Include dependency graph for searcharray.C:



### Defines

- #define **FACTOR** 1.2

*Spacing between elements as a factor of their size:.*

### Functions

- void **image\_patch** (const **Image**< **PixRGB**< **byte** > > &patch, const int ti, const int tj, **Image**< **PixRGB**< **byte** > > &image, const double alpha, **Image**< **byte** > &targets, bool do\_target)
- int **main** (const int argc, const char \*\*argv)

### 11.364.1 Detailed Description

create a randomized search array from two image patches

Definition in file [searcharray.C](#).

## 11.364.2 Define Documentation

### 11.364.2.1 `#define FACTOR 1.2`

Spacing between elements as a factor of their size:.

Definition at line 13 of file searcharray.C.

Referenced by `main()`.

## 11.364.3 Function Documentation

### 11.364.3.1 `int main (const int argc, const char ** argv)`

This program generates a randomized search array (and associated target mask) from two image patches (for target and distractors).

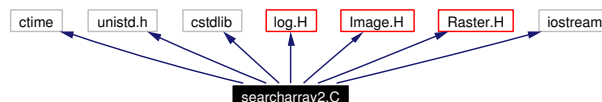
Definition at line 23 of file searcharray.C.

References `ASSERT`, `Image< PixRGB< float > >::beginw()`, `Image< PixRGB< float > >::endw()`, `FACTOR`, `Image< T >::getHeight()`, `Image< T >::getWidth()`, `image`, `main()`, `Raster::ReadRGB()`, `Raster::WriteGray()`, `Raster::WriteRGB()`, and `ZEROS`.

## 11.365 searcharray2.C File Reference

```
#include <ctime>
#include <unistd.h>
#include <cstdlib>
#include "log.H"
#include "Image.H"
#include "Raster.H"
#include <iostream>
```

Include dependency graph for searcharray2.C:



### Defines

- #define **FACTOR** 1.5

*Spacing between elements as a factor of their size:.*

### Functions

- void **image\_patch** (const **Image**< **PixRGB**< **byte** > > &patch, const int ti, const int tj, **Image**< **PixRGB**< **byte** > > &image, const double alpha, **Image**< **byte** > &targets, bool do\_target)
- int **main** (const int argc, const char \*\*argv)

### 11.365.1 Detailed Description

create a randomized search array from image patches

Definition in file [searcharray2.C](#).

## 11.365.2 Define Documentation

### 11.365.2.1 `#define FACTOR 1.5`

Spacing between elements as a factor of their size:.

Definition at line 47 of file searcharray2.C.

## 11.365.3 Function Documentation

### 11.365.3.1 `int main (const int argc, const char ** argv)`

This program generates a randomized search array (and associated target mask) from two image patches (for target and distractors).

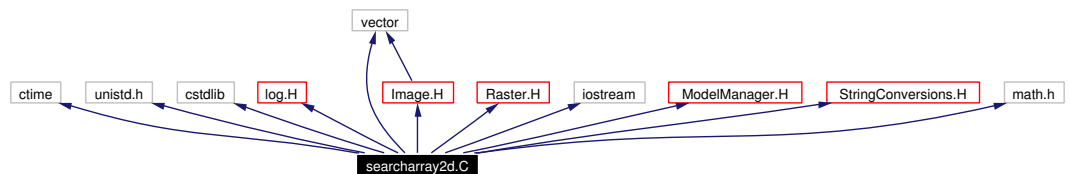
Definition at line 57 of file searcharray2.C.

References ASSERT, Image< PixRGB< float > >::beginw(), Image< PixRGB< float > >::endw(), FACTOR, Image< T >::getHeight(), Image< T >::getWidth(), image, main(), Raster::ReadRGB(), Raster::WriteGray(), Raster::WriteRGB(), and ZEROS.

## 11.366 searcharray2d.C File Reference

```
#include <ctime>
#include <unistd.h>
#include <cstdlib>
#include "log.H"
#include "Image.H"
#include "Raster.H"
#include <iostream>
#include "ModelManager.H"
#include <vector>
#include "StringConversions.H"
#include <math.h>
```

Include dependency graph for searcharray2d.C:



## Functions

- void **image\_patch** (const [Image](#)< [PixRGB](#)< [byte](#) > > &patch, const int ti, const int tj, [Image](#)< [PixRGB](#)< [byte](#) > > &image, const double alpha, const float FACTOR, [Image](#)< [byte](#) > &targets, bool do\_target, [Image](#)< [PixRGB](#)< [byte](#) > > &randomGrid, const [Image](#)< [PixRGB](#)< [byte](#) > > &randomNum)
- int [main](#) (const int argc, const char \*\*argv)

### 11.366.1 Detailed Description

create a randomized search array from image patches of a single target and the rest for distractors

Definition in file [searcharray2d.C](#).

## 11.366.2 Function Documentation

### 11.366.2.1 `int main (const int argc, const char ** argv)`

This program generates a randomized search array (and associated target mask) from several image patches (1 for target and the rest for distractors).

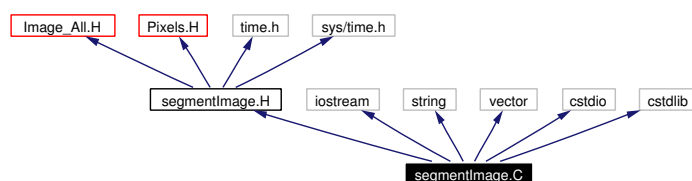
Definition at line 66 of file searcharray2d.C.

References `ASSERT`, `Image< PixRGB< float > >::beginw()`, `Image< PixRGB< float > >::endw()`, `FACTOR`, `ModelManager::getExtraArg()`, `image`, `main()`, `MYLOGVERB`, `ModelManager::numExtraArgs()`, `ModelManager::parseCommandLine()`, `randShuffle()`, `Raster::ReadRGB()`, `ModelComponent::start()`, `ModelComponent::stop()`, `toStr()`, `uint`, `Raster::WriteGray()`, `Raster::WriteRGB()`, and `ZEROS`.

## 11.367 segmentImage.C File Reference

```
#include "segmentImage.H"  
#include <iostream>  
#include <string>  
#include <vector>  
#include <cstdio>  
#include <cstdlib>
```

Include dependency graph for segmentImage.C:



### 11.367.1 Detailed Description

Basic image segmenter blob finder using color

Definition in file [segmentImage.C](#).



## 11.368 segmentImage.H File Reference

```
#include "Image_All.H"
```

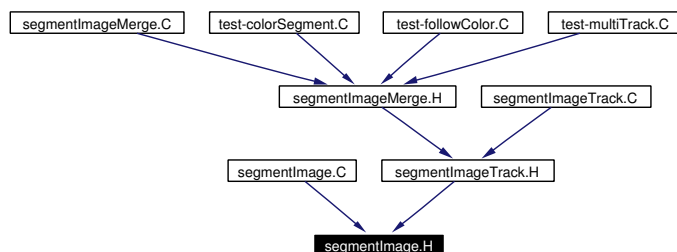
```
#include "Pixels.H"
```

```
#include <time.h>
```

```
#include <sys/time.h>
```

Include dependency graph for segmentImage.H:

This graph shows which files directly or indirectly include this file:



### Classes

- class **segmentImage**

### Defines

- #define **RGB** 1
- #define **HSV** 2
- #define **GREY** 3

#### 11.368.1 Detailed Description

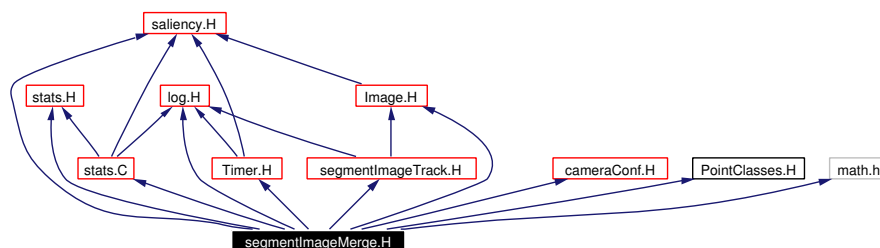
Basic image segmenter blob finder using color

Definition in file [segmentImage.H](#).

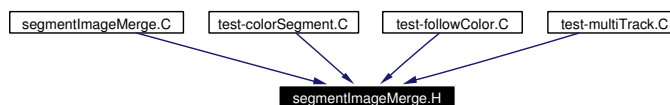
## 11.369 segmentImageMerge.H File Reference

```
#include "log.H"
#include "segmentImageTrack.H"
#include "Timer.H"
#include "saliency.H"
#include "Image.H"
#include "stats.H"
#include "stats.C"
#include "cameraConf.H"
#include "PointClasses.H"
#include <math.h>
```

Include dependency graph for segmentImageMerge.H:



This graph shows which files directly or indirectly include this file:



### Classes

- class **segmentImageMerge**

### 11.369.1 Detailed Description

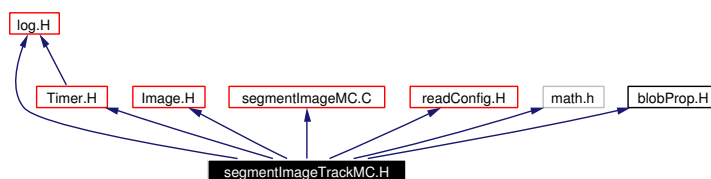
Basic image segmenter blob finder using color

Definition in file [segmentImageMerge.H](#).

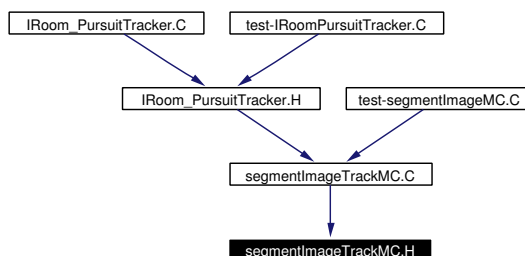
## 11.370 segmentImageTrackMC.H File Reference

```
#include "log.H"
#include "Timer.H"
#include "Image.H"
#include "segmentImageMC.C"
#include "readConfig.H"
#include <math.h>
#include "blobProp.H"
```

Include dependency graph for segmentImageTrackMC.H:



This graph shows which files directly or indirectly include this file:



### Classes

- class **segmentImageTrackMC**

### 11.370.1 Detailed Description

Basic image segmenter blob finder using color

Definition in file [segmentImageTrackMC.H](#).

## 11.371 serial.C File Reference

```
#include "serial.H"
```

```
#include "log.H"
```

```
#include <iostream>
```

```
#include "ModelManager.H"
```

Include dependency graph for serial.C:

### 11.371.1 Detailed Description

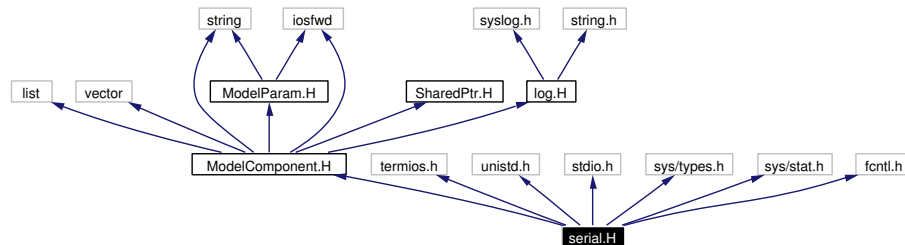
class for interfacing with a serial port

Definition in file [serial.C](#).

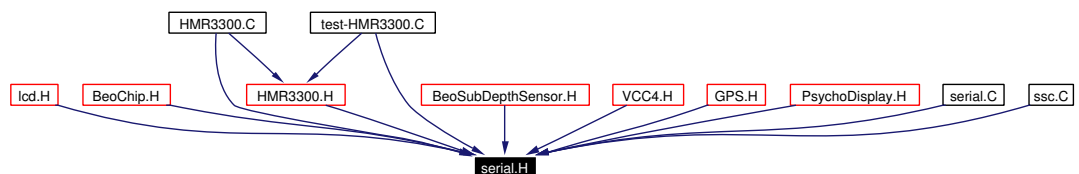
## 11.372 serial.H File Reference

```
#include "ModelComponent.H"
#include <termios.h>
#include <unistd.h>
#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
```

Include dependency graph for serial.H:



This graph shows which files directly or indirectly include this file:



## Classes

- class [Serial](#)  
*Interface to a serial port.*

## Enumerations

- enum [serialError](#) {

```
serialErrSuccess = 0, serialErrOpenNoTty, serialErrOpenFailed, serialErr-  
SpeedInvalid,  
serialErrFlowInvalid, serialErrParityInvalid, serialErrCharsizeInvalid,  
serialErrStopbitsInvalid,  
serialErrOptionInvalid, serialErrResourceFailure, serialErrOutput, serial-  
ErrInput,  
serialErrTimeout, serialErrExtended, serialErrReadFailed, serialErrRead-  
TimedOut,  
serialErrWriteFailed, serialErrFcntlFailed, serialErrTcGetAttrFailed,  
serialErrTcSetAttrFailed }
```

*enum for different serial errors that could occur*

### 11.372.1 Detailed Description

class for interfacing with a serial port

Definition in file [serial.H](#).

### 11.372.2 Enumeration Type Documentation

#### 11.372.2.1 enum [serialError](#)

enum for different serial errors that could occur

Definition at line 49 of file serial.H.

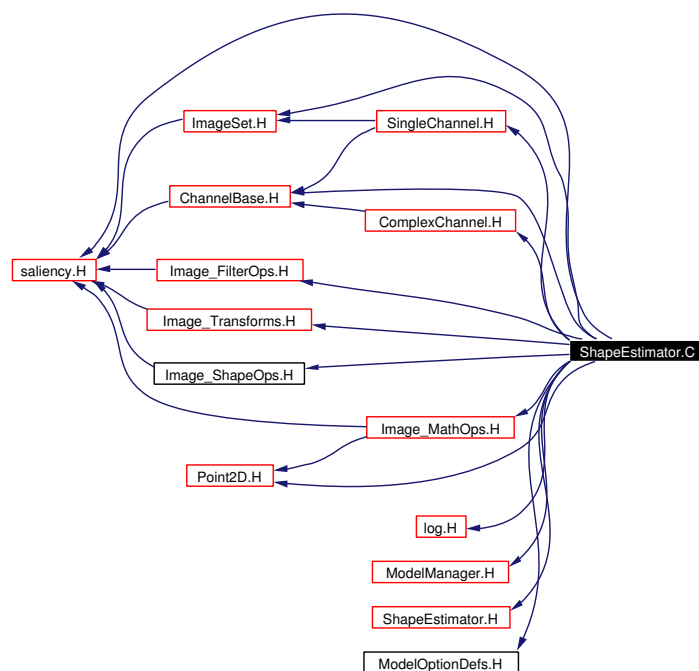
Referenced by `Serial::setBlocking()`, `Serial::setCharBits()`, `Serial::setFlowControl()`, `Serial::setParity()`, `Serial::setSpeed()`, and `Serial::setStopBits()`.

## 11.373 ShapeEstimator.C File Reference

```
#include "saliency.H"
#include "log.H"
#include "ModelManager.H"
#include "ShapeEstimator.H"
#include "Point2D.H"
#include "ChannelBase.H"
#include "Image_FilterOps.H"
#include "Image_Transforms.H"
#include "Image_MathOps.H"
#include "Image_ShapeOps.H"
#include "ImageSet.H"
#include "SingleChannel.H"
#include "ComplexChannel.H"
#include "ModelOptionDefs.H"
```

Include dependency graph for ShapeEstimator.C:





## Defines

- `#define SIGMA 20`
- `#define CLAMP 0.4`

### 11.373.1 Detailed Description

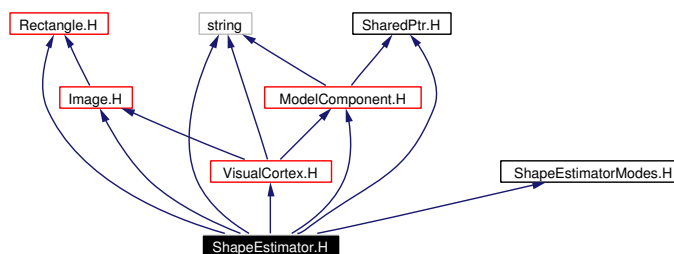
Estimate the shape/size of an attended object

Definition in file [ShapeEstimator.C](#).

## 11.374 ShapeEstimator.H File Reference

```
#include <string>
#include "Image.H"
#include "VisualCortex.H"
#include "SharedPtr.H"
#include "ModelComponent.H"
#include "ShapeEstimatorModes.H"
#include "Rectangle.H"
```

Include dependency graph for ShapeEstimator.H:



This graph shows which files directly or indirectly include this file:

### Classes

- class [ShapeEstimator](#)  
*everything realted to the shape estimation procedure*

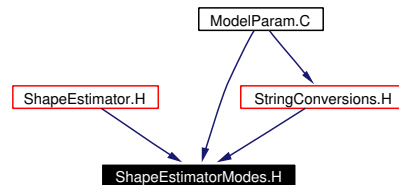
### 11.374.1 Detailed Description

Estimate the shape/size of an attended object

Definition in file [ShapeEstimator.H](#).

## 11.375 ShapeEstimatorModes.H File Reference

This graph shows which files directly or indirectly include this file:



### Defines

- #define `NBSHAPEESTIMATORMODES` 4  
number of known *ShapeEstimator* modes:
- #define `NBSHAPEESTIMATORSMOOTHMETHODS` 3  
number of known *ShapeEstimator* smooth methods

### Enumerations

- enum `ShapeEstimatorMode` { `SEMnone` = 0, `SEMfeatureMap` = 1, `SEMconspicuityMap` = 2, `SEMsaliencyMap` = 3 }  
Possible *ShapeEstimator* modes.
- enum `ShapeEstimatorSmoothMethod` { `SESMnone` = 0, `SESMgaussian` = 1, `SESMchamfer` = 2 }  
Possible *ShapeEstimator* smooth methods.

### Functions

- const char \* `shapeEstimatorModeName` (const `ShapeEstimatorMode` p)  
Returns name of given mode.
- const char \* `shapeEstimatorSmoothMethodName` (const `ShapeEstimatorSmoothMethod` p)  
Returns name of given smooth method.

### 11.375.1 Detailed Description

definition of the shape estimator modes

Definition in file [ShapeEstimatorModes.H](#).

### 11.375.2 Define Documentation

#### 11.375.2.1 `#define NBSHAPEESTIMATORMODES` 4

number of known [ShapeEstimator](#) modes:

Definition at line 50 of file [ShapeEstimatorModes.H](#).

Referenced by [convertFromString\(\)](#), and [shapeEstimatorModeName\(\)](#).

#### 11.375.2.2 `#define NBSHAPEESTIMATORSMOOTHMETHODS` 3

number of known [ShapeEstimator](#) smooth methods

Definition at line 70 of file [ShapeEstimatorModes.H](#).

Referenced by [convertFromString\(\)](#), and [shapeEstimatorSmoothMethodName\(\)](#).

### 11.375.3 Enumeration Type Documentation

#### 11.375.3.1 `enum ShapeEstimatorMode`

Possible [ShapeEstimator](#) modes.

##### Enumeration values:

***SEMfeatureMap*** No shape estimation.

***SEMconspicuityMap*** Based on winning feature map.

***SEMsaliencyMap*** Based on winning conspicuity map.

Definition at line 41 of file [ShapeEstimatorModes.H](#).

Referenced by [convertFromString\(\)](#), [ShapeEstimator::ShapeEstimator\(\)](#), and [Brain::start1\(\)](#).

#### 11.375.3.2 `enum ShapeEstimatorSmoothMethod`

Possible [ShapeEstimator](#) smooth methods.

**Enumeration values:**

***SESMgaussian*** Don't smooth at all.

***SESMchamfer*** smooth by convolving with a Gaussian

Definition at line 62 of file ShapeEstimatorModes.H.

Referenced by convertFromString(), and ShapeEstimator::ShapeEstimator().

**11.375.4 Function Documentation****11.375.4.1 const char\* shapeEstimatorModeName (const [ShapeEstimatorMode](#) p) [inline]**

Returns name of given mode.

Definition at line 53 of file ShapeEstimatorModes.H.

References NBSHAPEESTIMATORMODES.

Referenced by convertFromString(), and convertToString().

**11.375.4.2 const char\* shapeEstimatorSmoothMethodName (const [ShapeEstimatorSmoothMethod](#) p) [inline]**

Returns name of given smooth method.

Definition at line 73 of file ShapeEstimatorModes.H.

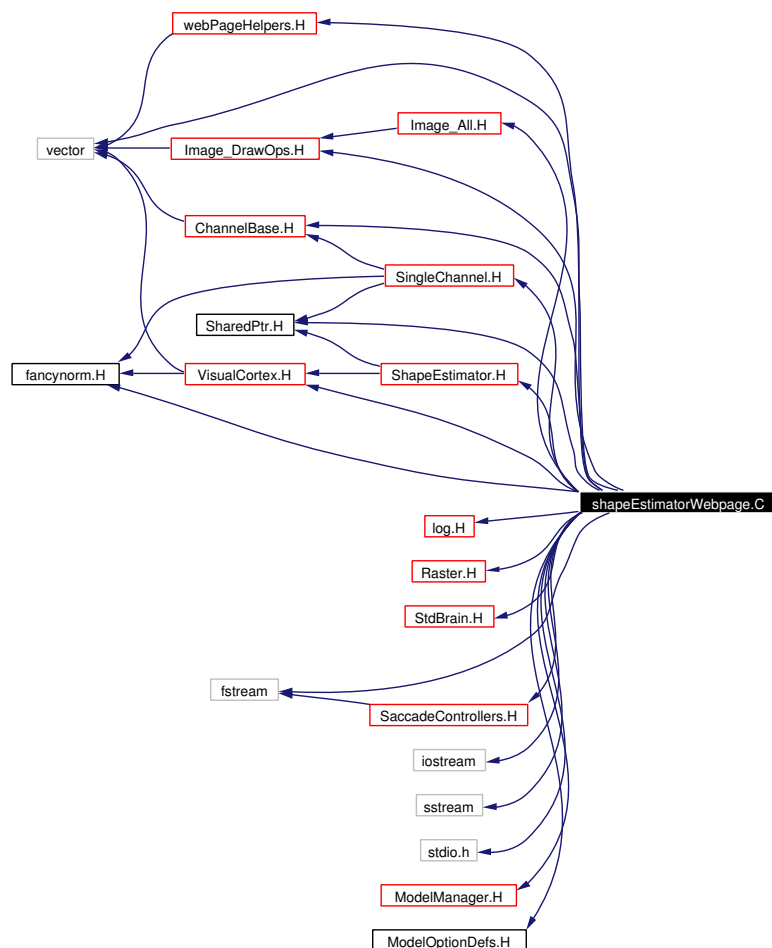
References NBSHAPEESTIMATORSMOOTHMETHODS.

Referenced by convertFromString(), and convertToString().

## 11.376 shapeEstimatorWebpage.C File Reference

```
#include "Image_All.H"
#include "Image_DrawOps.H"
#include "log.H"
#include "Raster.H"
#include "StdBrain.H"
#include "VisualCortex.H"
#include "SaccadeControllers.H"
#include "ChannelBase.H"
#include "SharedPtr.H"
#include "fancynorm.H"
#include "webPageHelpers.H"
#include <iostream>
#include <sstream>
#include <fstream>
#include <vector>
#include <stdio.h>
#include "SingleChannel.H"
#include "ModelManager.H"
#include "ShapeEstimator.H"
#include "ModelOptionDefs.H"
```

Include dependency graph for shapeEstimatorWebpage.C:



## Functions

- int **main** (int argc, char \*\*argv)

### 11.376.1 Detailed Description

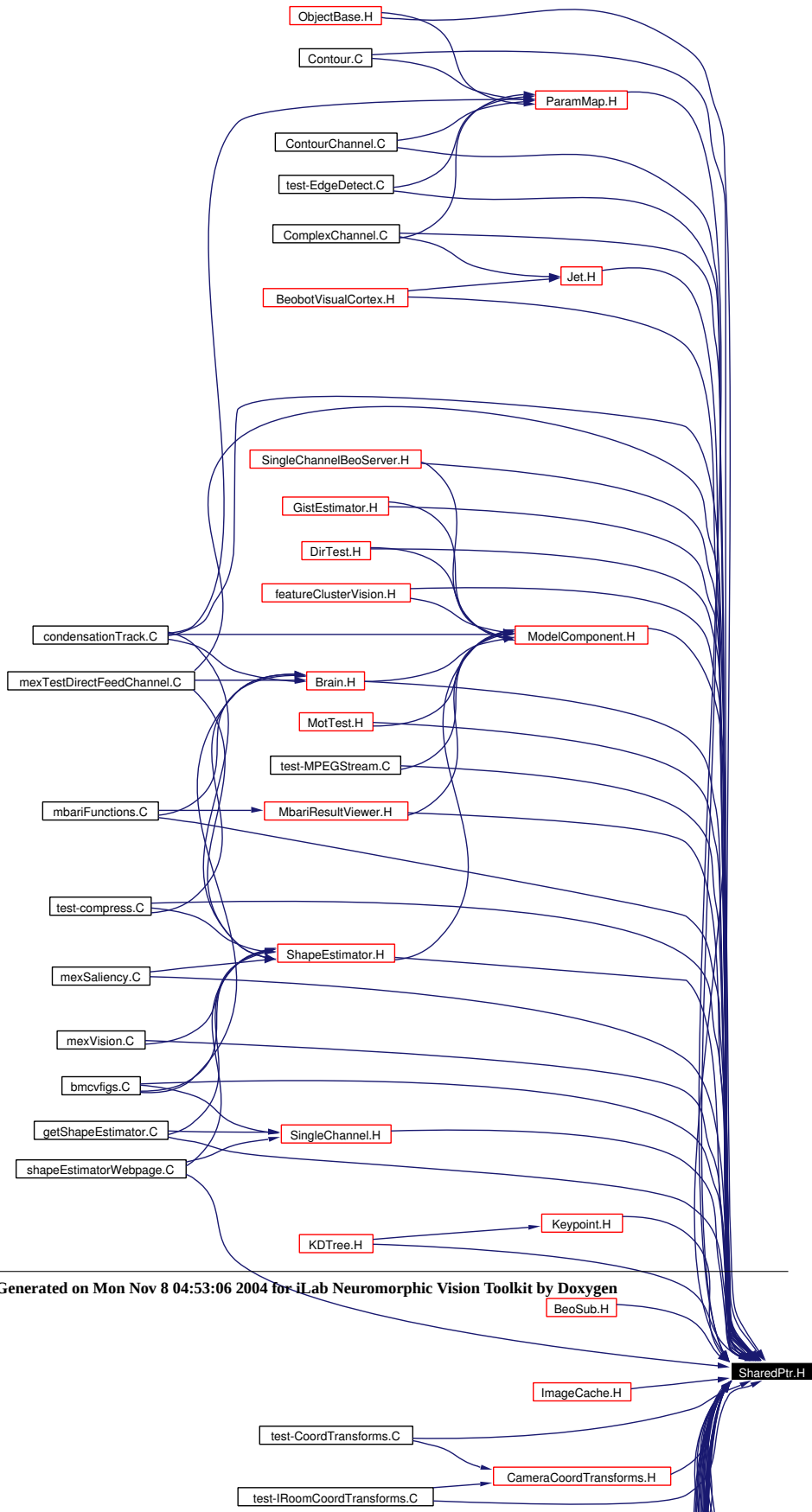
make web page(s) for SE demonstration

Definition in file [shapeEstimatorWebpage.C](#).

## 11.377 SharedPtr.H File Reference

This graph shows which files directly or indirectly include this file:





## Classes

- class [SharedPtr](#)

*A smart pointer with reference counted copy semantics.*

## Functions

- template<class T, class U> bool [operator==](#) (const [SharedPtr](#)< T > &a, const [SharedPtr](#)< U > &b)

*Test whether two SharedPtr's point to the same object.*

- template<class T, class U> bool [operator!=](#) (const [SharedPtr](#)< T > &a, const [SharedPtr](#)< U > &b)

*Test whether two SharedPtr's point to different objects.*

- template<class T> [SharedPtr](#)< T > [makeShared](#) (T \*t)

*A convenience function for making a [SharedPtr](#) out of a raw pointer.*

### 11.377.1 Detailed Description

a shared pointer class

Definition in file [SharedPtr.H](#).

### 11.377.2 Function Documentation

#### 11.377.2.1 template<class T> [SharedPtr](#)<T> [makeShared](#) (T \* t)

A convenience function for making a [SharedPtr](#) out of a raw pointer.

Definition at line 182 of file [SharedPtr.H](#).

Referenced by [VisualCortexSurprise::addDefaultChannels\(\)](#), [VisualCortexBeo::addDefaultChannels\(\)](#), [VisualCortex::addDefaultChannels\(\)](#), [BlueChannel::BlueChannel\(\)](#), [BlueYellowChannel::BlueYellowChannel\(\)](#), [SoxChannel::buildSubChans\(\)](#), [EndPointChannelSurprise::buildSubChans\(\)](#), [XJunctionChannelSurprise::buildSubChans\(\)](#), [TJunctionChannelSurprise::buildSubChans\(\)](#), [LJunctionChannelSurprise::buildSubChans\(\)](#), [MotionChannelSurprise::buildSubChans\(\)](#), [MultiColorBandChannelSurprise::buildSubChans\(\)](#), [OrientationChannelSurprise::buildSubChans\(\)](#), [StereoChannel::buildSubChans\(\)](#), [EndPointChannel::buildSubChans\(\)](#), [XJunctionChannel::buildSubChans\(\)](#), [TJunctionChannel::buildSubChans\(\)](#), [LJunctionChannel::buildSubChans\(\)](#), [MotionChannel::buildSubChans\(\)](#),

MultiColorBandChannel::buildSubChans(), OrientationChannel::buildSubChans(), ColorBandChannel::ColorBandChannel(), ColorChannel::ColorChannel(), ColorChannelSurprise::ColorChannelSurprise(), CompositeColorChannel::CompositeColorChannel(), CompositeColorChannelSurprise::CompositeColorChannelSurprise(), KDTree::createKDTree(), DirectionChannel::DirectionChannel(), FlickerChannel::FlickerChannel(), GreenChannel::GreenChannel(), HueChannel::HueChannel(), BeobotVisualCortex::init(), IntensityChannel::IntensityChannel(), RedChannel::RedChannel(), RedGreenChannel::RedGreenChannel(), SimulationViewerStd::saveResults(), ConvolveChannel::setFilter(), RGBConvolveChannel::setFilters(), MultiDirectFeedChannel::setNumChans(), SOColorChannel::SOColorChannel(), SOColorChannelSurprise::SOColorChannelSurprise(), TemplateMatchChannel::TemplateMatchChannel(), and YellowChannel::YellowChannel().

#### 11.377.2.2 **template<class T, class U> bool operator!= (const SharedPtr< T > & a, const SharedPtr< U > & b) [inline]**

Test whether two SharedPtr's point to different objects.

Definition at line 175 of file SharedPtr.H.

References SharedPtr< T >::get().

#### 11.377.2.3 **template<class T, class U> bool operator== (const SharedPtr< T > & a, const SharedPtr< U > & b) [inline]**

Test whether two SharedPtr's point to the same object.

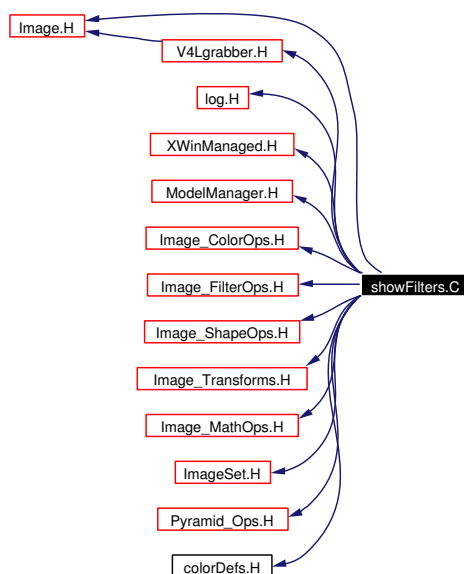
Definition at line 168 of file SharedPtr.H.

References SharedPtr< T >::get().

## 11.378 showFilters.C File Reference

```
#include "Image.H"  
#include "log.H"  
#include "V4Lgrabber.H"  
#include "XWinManaged.H"  
#include "ModelManager.H"  
#include "Image_ColorOps.H"  
#include "Image_FilterOps.H"  
#include "Image_ShapeOps.H"  
#include "Image_Transforms.H"  
#include "Image_MathOps.H"  
#include "ImageSet.H"  
#include "Pyramid_Ops.H"  
#include "colorDefs.H"
```

Include dependency graph for showFilters.C:



## Functions

- int [main](#) (const int argc, const char \*\*argv)

### 11.378.1 Detailed Description

Import images from a video device and display a number of filters

Definition in file [showFilters.C](#).

### 11.378.2 Function Documentation

#### 11.378.2.1 int main (const int *argc*, const char \*\* *argv*)

This program grabs a frame from a framegrabber and computes and displays a number of low-level features. This was done as a demo to explain the dissection of an image into features for a public outreach project.

Definition at line 60 of file showFilters.C.

References [absDiff\(\)](#), [CloseButtonListener::add\(\)](#), [buildPyrGaussian\(\)](#), [colorStain\(\)](#), [XWinManaged::drawImage\(\)](#), [FULLTRACE](#), [SharedPtr< T >::get\(\)](#), [Image< T >::getDims\(\)](#), [getRGBY\(\)](#), [luminance\(\)](#), [main\(\)](#), [makeBinary\(\)](#), [Image< T >::normalize\(\)](#), [CloseButtonListener::pressedAnyCloseButton\(\)](#), [rescale\(\)](#), and [takeMax\(\)](#).

## 11.379 SimulationViewer.C File Reference

```
#include "SimulationViewer.H"  
#include "ModelManager.H"  
#include "ModelOptionDefs.H"  
#include "SimulationViewerStd.H"  
#include "SimulationViewerCompress.H"  
#include "SimulationViewerEyeMvt.H"  
#include "SimulationViewerEyeMvt2.H"
```

Include dependency graph for SimulationViewer.C:

### 11.379.1 Detailed Description

visualize various model simulations

Definition in file [SimulationViewer.C](#).

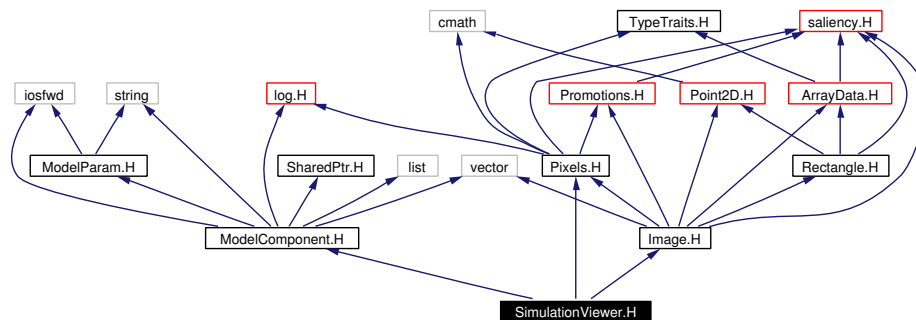
## 11.380 SimulationViewer.H File Reference

```
#include "ModelComponent.H"
```

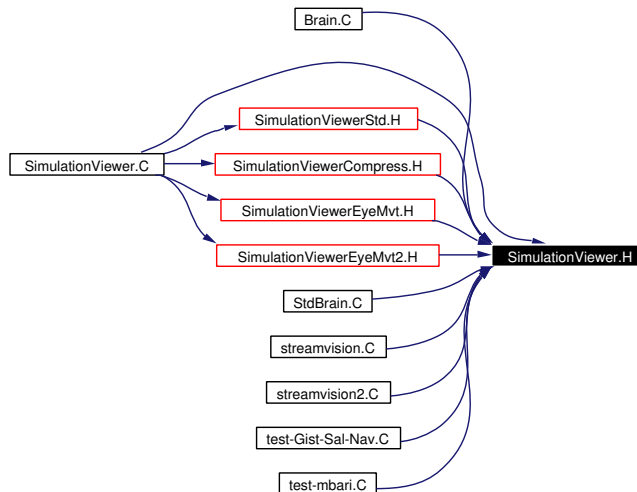
```
#include "Pixels.H"
```

```
#include "Image.H"
```

Include dependency graph for SimulationViewer.H:



This graph shows which files directly or indirectly include this file:



### Classes

- class [SimulationViewer](#)

*This class provides an abstract interface for a simulation viewer.*

- class [SimulationViewerConfigurator](#)  
*SimulationViewer* configurator.

### 11.380.1 Detailed Description

visualize various model simulations

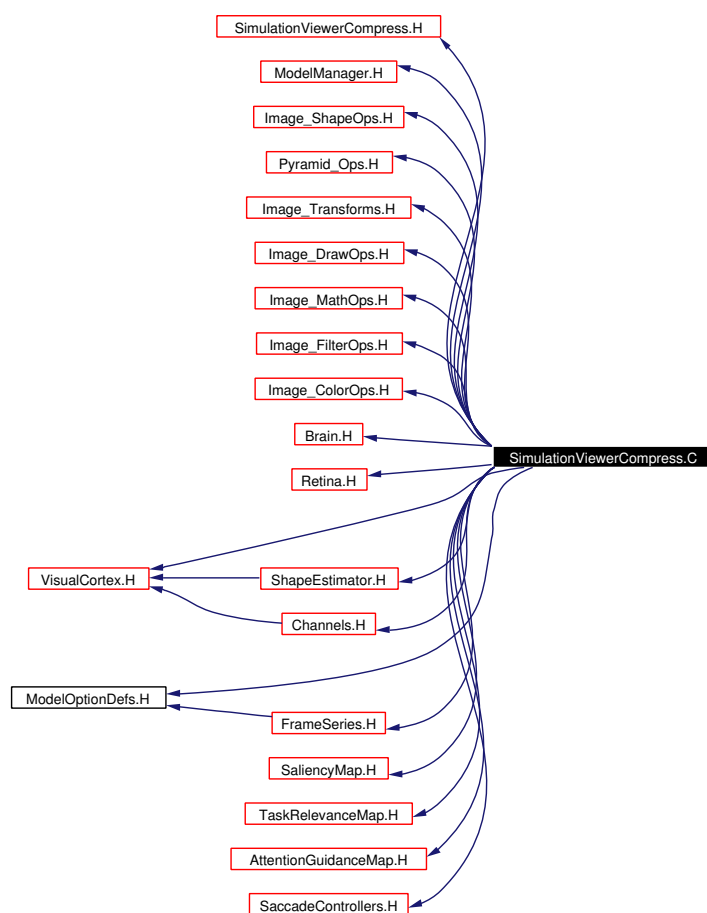
Definition in file [SimulationViewer.H](#).



## 11.381 SimulationViewerCompress.C File Reference

```
#include "SimulationViewerCompress.H"
#include "ModelManager.H"
#include "Image_ShapeOps.H"
#include "Pyramid_Ops.H"
#include "Image_Transforms.H"
#include "Image_DrawOps.H"
#include "Image_MathOps.H"
#include "Image_FilterOps.H"
#include "Image_ColorOps.H"
#include "Brain.H"
#include "Retina.H"
#include "VisualCortex.H"
#include "FrameSeries.H"
#include "ModelOptionDefs.H"
#include "SaliencyMap.H"
#include "TaskRelevanceMap.H"
#include "AttentionGuidanceMap.H"
#include "SaccadeControllers.H"
#include "ShapeEstimator.H"
#include "Channels.H"
```

Include dependency graph for SimulationViewerCompress.C:



## Defines

- #define [NRND](#) 1

*Number of random samples to take.*

### 11.381.1 Detailed Description

multi-foveated saliency-based compression

Definition in file [SimulationViewerCompress.C](#).

## 11.381.2 Define Documentation

### 11.381.2.1 `#define NRND 1`

Number of random samples to take.

Definition at line 61 of file SimulationViewerCompress.C.

Referenced by SimulationViewerCompress::getTraj().

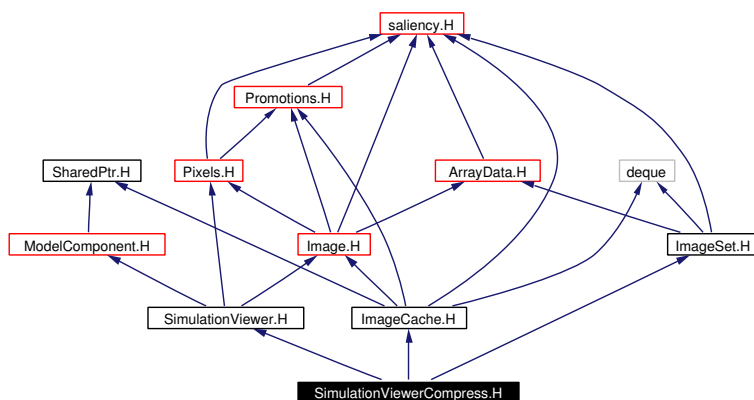
## 11.382 SimulationViewerCompress.H File Reference

```
#include "SimulationViewer.H"
```

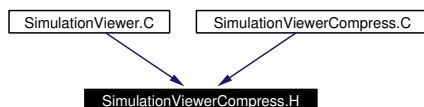
```
#include "ImageSet.H"
```

```
#include "ImageCache.H"
```

Include dependency graph for SimulationViewerCompress.H:



This graph shows which files directly or indirectly include this file:



### Classes

- class [SimulationViewerCompress](#)  
*Do multi-foveated saliency-based image compression.*

### 11.382.1 Detailed Description

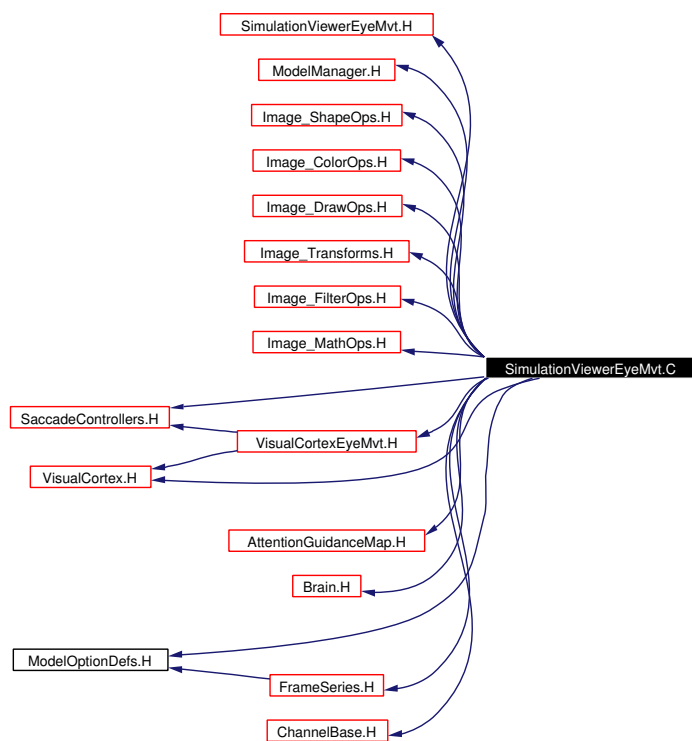
multi-foveated saliency-based compression

Definition in file [SimulationViewerCompress.H](#).

## 11.383 SimulationViewerEyeMvt.C File Reference

```
#include "SimulationViewerEyeMvt.H"
#include "ModelManager.H"
#include "Image_ShapeOps.H"
#include "Image_ColorOps.H"
#include "Image_DrawOps.H"
#include "Image_Transforms.H"
#include "Image_FilterOps.H"
#include "Image_MathOps.H"
#include "SaccadeControllers.H"
#include "AttentionGuidanceMap.H"
#include "Brain.H"
#include "FrameSeries.H"
#include "ModelOptionDefs.H"
#include "VisualCortex.H"
#include "ChannelBase.H"
#include "VisualCortexEyeMvt.H"
```

Include dependency graph for SimulationViewerEyeMvt.C:



## Defines

- `#define NBRND 100`

### 11.383.1 Detailed Description

comparison between saliency and human eye movements

Definition in file [SimulationViewerEyeMvt.C](#).

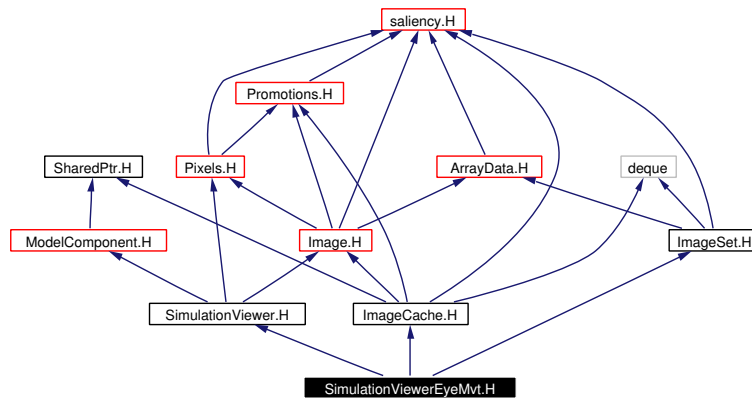
## 11.384 SimulationViewerEyeMvt.H File Reference

```
#include "SimulationViewer.H"
```

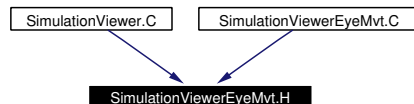
```
#include "ImageSet.H"
```

```
#include "ImageCache.H"
```

Include dependency graph for SimulationViewerEyeMvt.H:



This graph shows which files directly or indirectly include this file:



### Classes

- class [SimulationViewerEyeMvt](#)  
*Measure salience at human eye positions.*

#### 11.384.1 Detailed Description

comparison between saliency and human eye movements

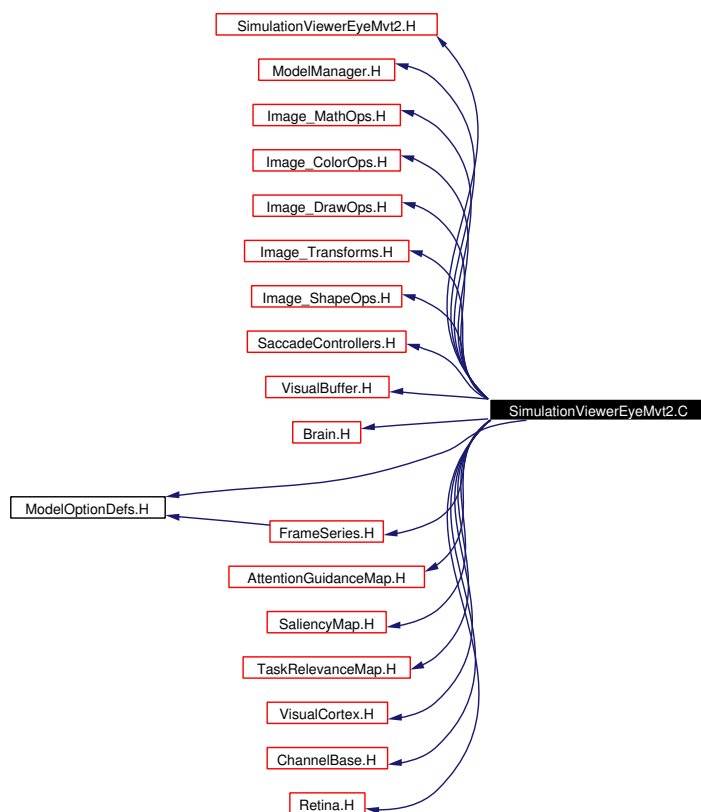
Definition in file [SimulationViewerEyeMvt.H](#).

## 11.385 SimulationViewerEyeMvt2.C File Reference

```
#include "SimulationViewerEyeMvt2.H"
#include "ModelManager.H"
#include "Image_MathOps.H"
#include "Image_ColorOps.H"
#include "Image_DrawOps.H"
#include "Image_Transforms.H"
#include "Image_ShapeOps.H"
#include "SaccadeControllers.H"
#include "VisualBuffer.H"
#include "Brain.H"
#include "FrameSeries.H"
#include "ModelOptionDefs.H"
#include "AttentionGuidanceMap.H"
#include "SaliencyMap.H"
#include "TaskRelevanceMap.H"
#include "VisualCortex.H"
#include "ChannelBase.H"
#include "Retina.H"
```

Include dependency graph for SimulationViewerEyeMvt2.C:





### 11.385.1 Detailed Description

comparison between saliency and human eye movements

Definition in file [SimulationViewerEyeMvt2.C](#).

## 11.386 SimulationViewerEyeMvt2.H File Reference

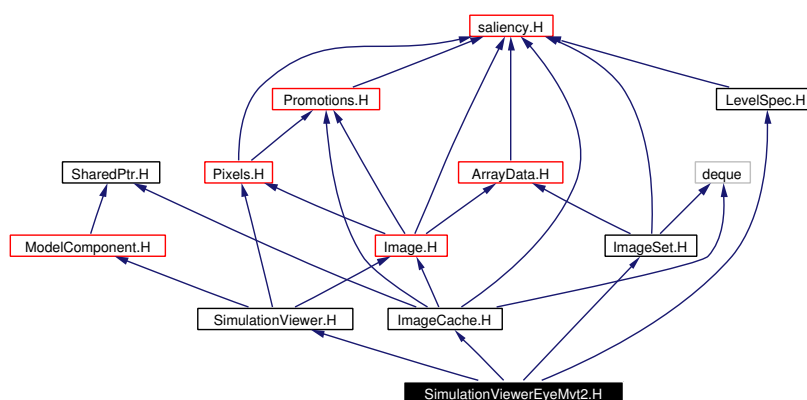
```
#include "SimulationViewer.H"
```

```
#include "ImageSet.H"
```

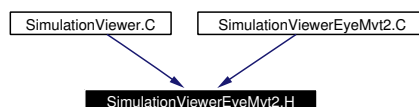
```
#include "ImageCache.H"
```

```
#include "LevelSpec.H"
```

Include dependency graph for SimulationViewerEyeMvt2.H:



This graph shows which files directly or indirectly include this file:



### Classes

- class [SimulationViewerEyeMvt2](#)  
*Measure salience at human eye positions.*

### 11.386.1 Detailed Description

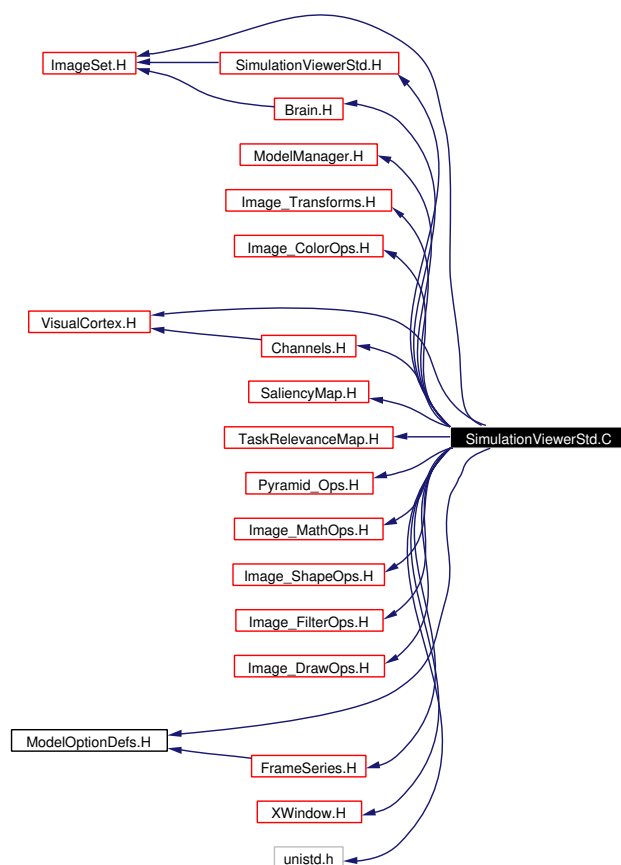
comparison between saliency and human eye movements

Definition in file [SimulationViewerEyeMvt2.H](#).

## 11.387 SimulationViewerStd.C File Reference

```
#include "SimulationViewerStd.H"
#include "ModelManager.H"
#include "Image_Transforms.H"
#include "Image_ColorOps.H"
#include "VisualCortex.H"
#include "SaliencyMap.H"
#include "TaskRelevanceMap.H"
#include "ImageSet.H"
#include "Pyramid_Ops.H"
#include "Image_MathOps.H"
#include "Image_ShapeOps.H"
#include "Image_FilterOps.H"
#include "Image_DrawOps.H"
#include "Brain.H"
#include "FrameSeries.H"
#include "XWindow.H"
#include "Channels.H"
#include "ModelOptionDefs.H"
#include <unistd.h>
```

Include dependency graph for SimulationViewerStd.C:



### 11.387.1 Detailed Description

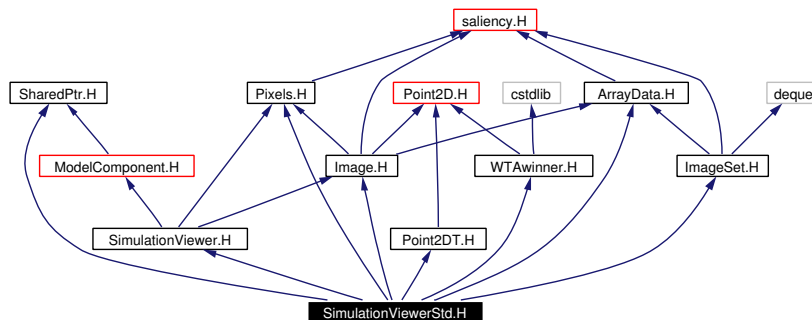
visualize various model simulations

Definition in file [SimulationViewerStd.C](#).

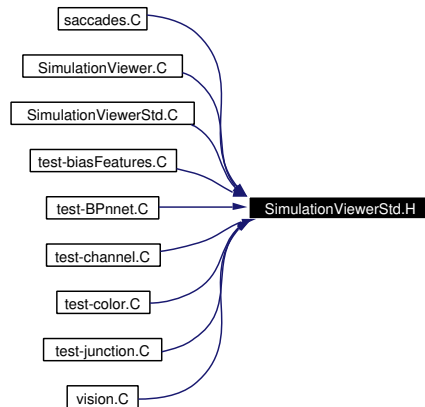
## 11.388 SimulationViewerStd.H File Reference

```
#include "SimulationViewer.H"
#include "Pixels.H"
#include "Image.H"
#include "ArrayData.H"
#include "SharedPtr.H"
#include "Point2DT.H"
#include "WTAwinner.H"
#include "ImageSet.H"
```

Include dependency graph for SimulationViewerStd.H:



This graph shows which files directly or indirectly include this file:



## Classes

- class [SimulationViewerStd](#)

*This class provides a standard set of visualization tools.*

### 11.388.1 Detailed Description

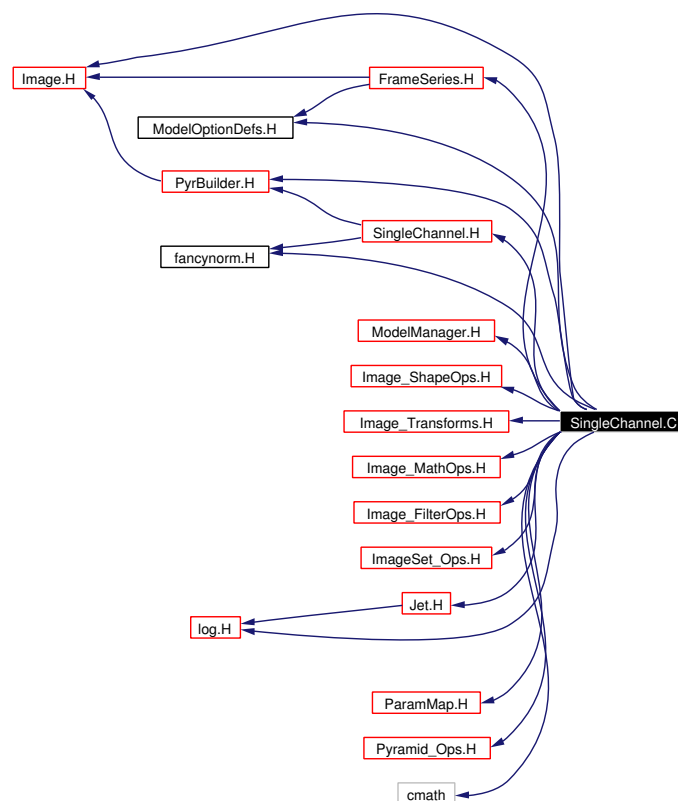
visualize various model simulations

Definition in file [SimulationViewerStd.H](#).

## 11.389 SingleChannel.C File Reference

```
#include "SingleChannel.H"
#include "ModelManager.H"
#include "fancynorm.H"
#include "Image.H"
#include "Image_ShapeOps.H"
#include "Image_Transforms.H"
#include "Image_MathOps.H"
#include "Image_FilterOps.H"
#include "ImageSet_Ops.H"
#include "Jet.H"
#include "log.H"
#include "ParamMap.H"
#include "Pyramid_Ops.H"
#include "PyrBuilder.H"
#include "FrameSeries.H"
#include "ModelOptionDefs.H"
#include <cmath>
```

Include dependency graph for SingleChannel.C:



### 11.389.1 Detailed Description

Channel for a single stream of processing.

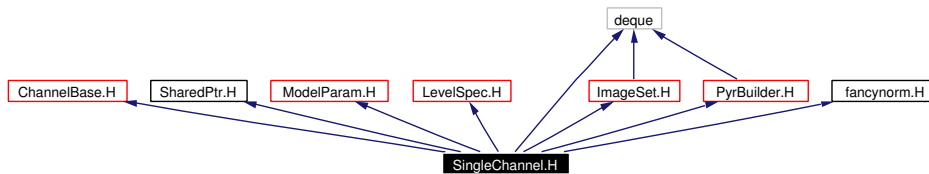
Definition in file [SingleChannel.C](#).



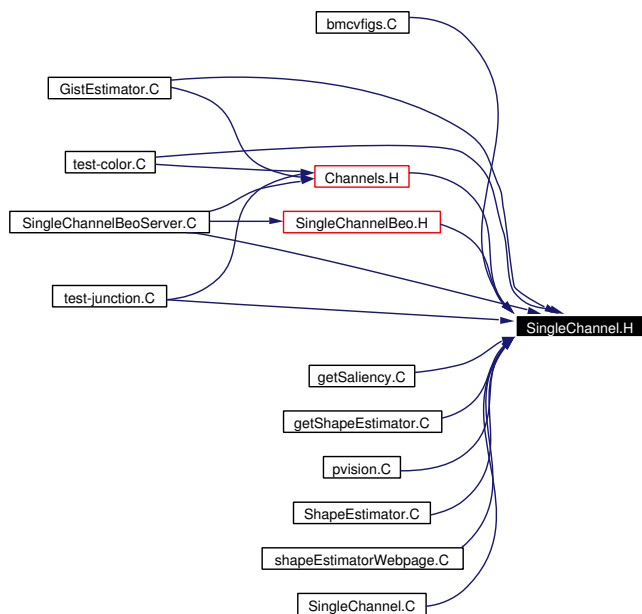
## 11.390 SingleChannel.H File Reference

```
#include "ChannelBase.H"  
#include "SharedPtr.H"  
#include "ModelParam.H"  
#include "LevelSpec.H"  
#include "ImageSet.H"  
#include "PyrBuilder.H"  
#include "fancynorm.H"  
#include <deque>
```

Include dependency graph for SingleChannel.H:



This graph shows which files directly or indirectly include this file:



## Classes

- class [SingleChannel](#)  
*SingleChannel represents a single stream of processing.*
- struct **SingleChannel::TPyr**

### 11.390.1 Detailed Description

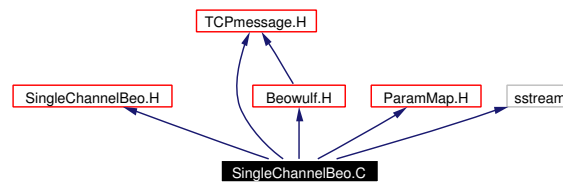
Channel for a single stream of processing.

Definition in file [SingleChannel.H](#).

## 11.391 SingleChannelBeo.C File Reference

```
#include "SingleChannelBeo.H"  
#include "TCPmessage.H"  
#include "Beowulf.H"  
#include "ParamMap.H"  
#include <sstream>
```

Include dependency graph for SingleChannelBeo.C:



### 11.391.1 Detailed Description

wrapper class to run a [SingleChannel](#) on a distant CPU

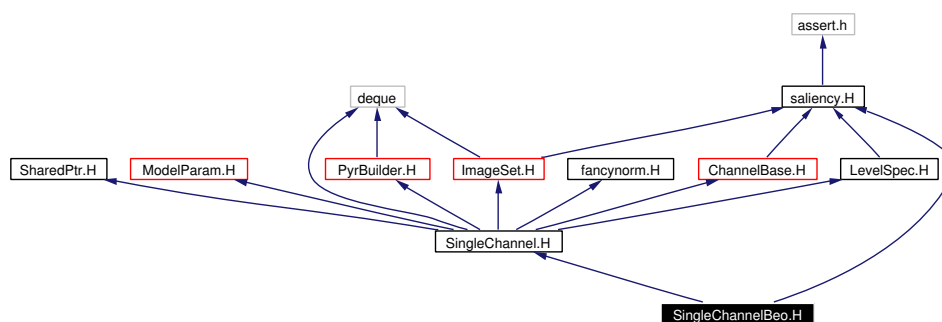
Definition in file [SingleChannelBeo.C](#).

## 11.392 SingleChannelBeo.H File Reference

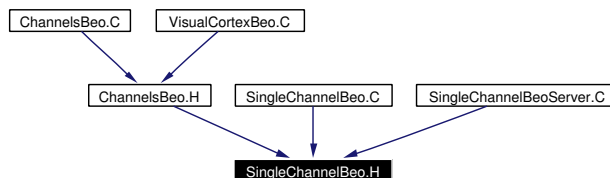
```
#include "saliency.H"
```

```
#include "SingleChannel.H"
```

Include dependency graph for SingleChannelBeo.H:



This graph shows which files directly or indirectly include this file:



## Classes

- class [SingleChannelBeo](#)  
Wrapper to run a [SingleChannel](#) on a distant CPU.

## Defines

- #define **BEO\_SCHANCONF** 1000
- #define **BEO\_SCHANINPUT** 1001
- #define **BEO\_SCHANOUTPUT** 1002
- #define **BEO\_SCHANALLOUT** 1003

### 11.392.1 Detailed Description

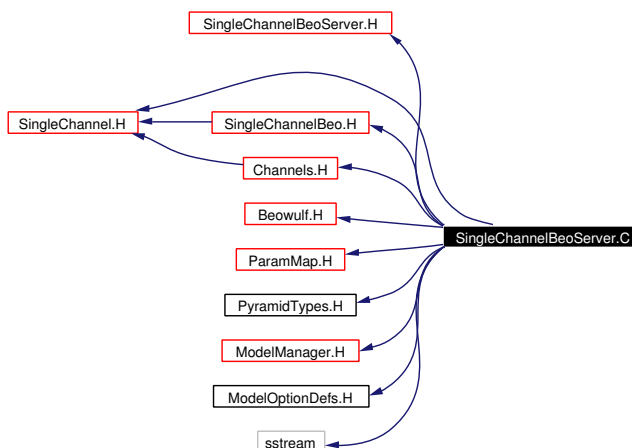
wrapper class to run a [SingleChannel](#) on a distant CPU

Definition in file [SingleChannelBeo.H](#).

## 11.393 SingleChannelBeoServer.C File Reference

```
#include "SingleChannelBeoServer.H"
#include "SingleChannelBeo.H"
#include "SingleChannel.H"
#include "Beowulf.H"
#include "ParamMap.H"
#include "Channels.H"
#include "PyramidTypes.H"
#include "ModelManager.H"
#include "ModelOptionDefs.H"
#include <sstream>
```

Include dependency graph for SingleChannelBeoServer.C:



### 11.393.1 Detailed Description

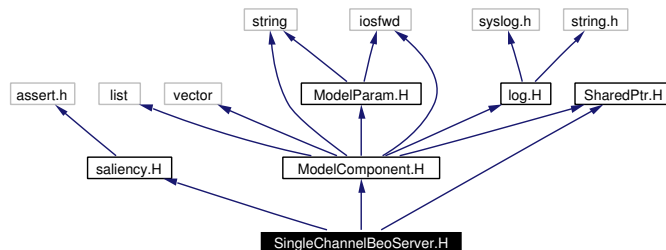
server to work with [SingleChannelBeo](#)

Definition in file [SingleChannelBeoServer.C](#).

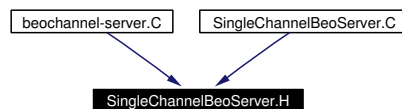
## 11.394 SingleChannelBeoServer.H File Reference

```
#include "saliency.H"
#include "ModelComponent.H"
#include "SharedPtr.H"
```

Include dependency graph for SingleChannelBeoServer.H:



This graph shows which files directly or indirectly include this file:



### Classes

- class [SingleChannelBeoServer](#)  
Server to work with [SingleChannelBeo](#).

### 11.394.1 Detailed Description

server to work with [SingleChannelBeo](#)

Definition in file [SingleChannelBeoServer.H](#).

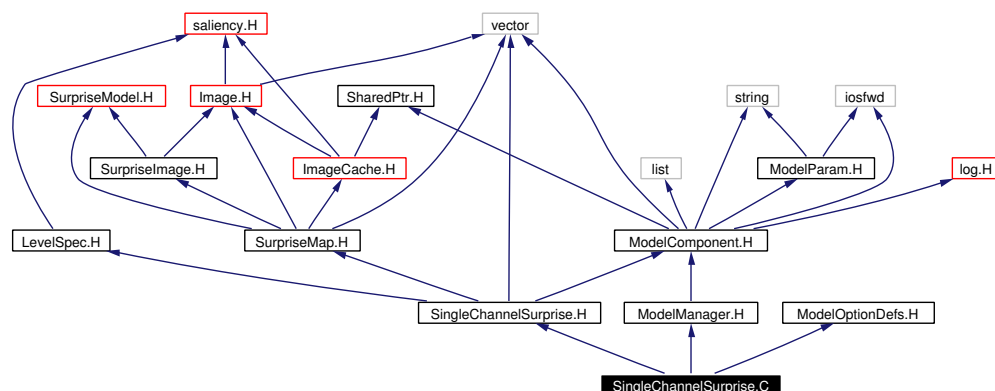
## 11.395 SingleChannelSurprise.C File Reference

```
#include "SingleChannelSurprise.H"
```

```
#include "ModelManager.H"
```

```
#include "ModelOptionDefs.H"
```

Include dependency graph for SingleChannelSurprise.C:



### 11.395.1 Detailed Description

Channel for a single stream of processing.

Definition in file [SingleChannelSurprise.C](#).

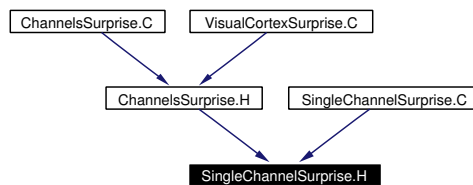


## 11.396 SingleChannelSurprise.H File Reference

```
#include "ModelComponent.H"  
#include "SurpriseMap.H"  
#include <vector>  
#include "LevelSpec.H"
```

Include dependency graph for SingleChannelSurprise.H:

This graph shows which files directly or indirectly include this file:



### Classes

- class [SingleChannelSurprise](#)  
*SingleChannelSurprise is a Surprise extension to [SingleChannel](#).*

### 11.396.1 Detailed Description

Channel for a single stream of processing.

Definition in file [SingleChannelSurprise.H](#).

## 11.397 SockServ.C File Reference

```
#include "TCPdefs.H"
#include "log.H"
#include "ModelOptionDefs.H"
#include "ModelManager.H"
#include "SockServ.H"
#include <sys/types.h>
#include <sys/socket.h>
#include <sys/time.h>
#include <stdio.h>
#include <unistd.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <errno.h>
#include <string.h>
#include <ctype.h>
#include <netdb.h>
#include <time.h>
```

Include dependency graph for SockServ.C:

### Defines

- #define **MYLOGID** sock
- #define **SOCKSERV\_UNCONFIGURED** 0  
*internal states (in 'state' member):*
- #define **SOCKSERV\_RUNNING** 1
- #define **SOCKSERV\_BOGUS** 2
- #define **CLITIMEOUT** 3000  
*timeout for termination of idle clients, in seconds:*

### 11.397.1 Detailed Description

A simple multi-client socket server

Definition in file [SockServ.C](#).

## 11.397.2 Define Documentation

### 11.397.2.1 `#define CLITIMEOUT 3000`

timeout for termination of idle clients, in seconds:

Definition at line 64 of file SockServ.C.

Referenced by SockServ::check().

### 11.397.2.2 `#define SOCKSERV_UNCONFIGURED 0`

internal states (in 'state' member):

Definition at line 59 of file SockServ.C.

Referenced by SockServ::SockServ().

## 11.398 SockServ.H File Reference

```
#include "ModelComponent.H"
```

```
#include <sys/time.h>
```

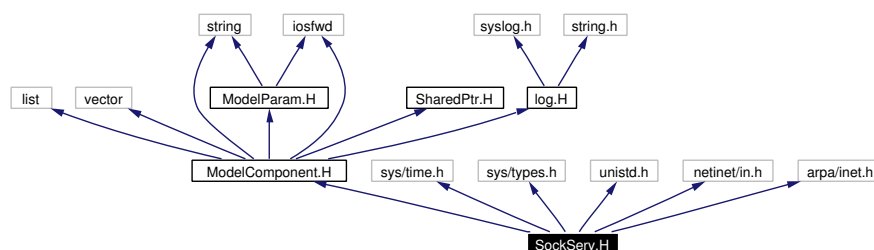
```
#include <sys/types.h>
```

```
#include <unistd.h>
```

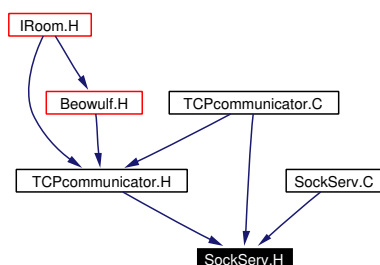
```
#include <netinet/in.h>
```

```
#include <arpa/inet.h>
```

Include dependency graph for SockServ.H:



This graph shows which files directly or indirectly include this file:



## Classes

- class [SockServ](#)

*This is a multi-socket connection server.*

## Defines

- #define [MAXNBNUCLI](#) 128  
*max number of new clients that can come between two check():*
- #define [SOCKSERV\\_IDLE](#) 0  
*return values for check():*
- #define [SOCKSERV\\_BUG](#) 1
- #define [SOCKSERV\\_ACTIV](#) 2

### 11.398.1 Detailed Description

A simple multi-client socket server

Definition in file [SockServ.H](#).

### 11.398.2 Define Documentation

#### 11.398.2.1 #define MAXNBNUCLI 128

max number of new clients that can come between two check():

Definition at line 49 of file SockServ.H.

Referenced by SockServ::addUserClient().

#### 11.398.2.2 #define SOCKSERV\_IDLE 0

return values for check():

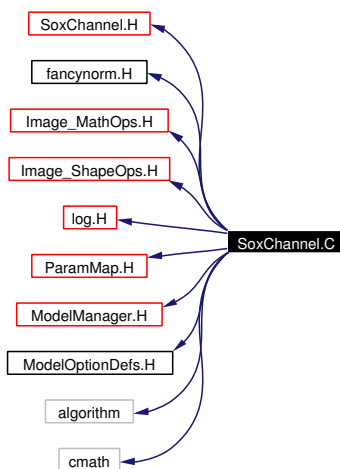
Definition at line 52 of file SockServ.H.

Referenced by SockServ::check(), and TCPcommunicator::run().

## 11.399 SoxChannel.C File Reference

```
#include "SoxChannel.H"  
#include "fancynorm.H"  
#include "Image_MathOps.H"  
#include "Image_ShapeOps.H"  
#include "log.H"  
#include "ParamMap.H"  
#include "ModelManager.H"  
#include "ModelOptionDefs.H"  
#include <algorithm>  
#include <cmath>
```

Include dependency graph for SoxChannel.C:



### 11.399.1 Detailed Description

Shortrange Orientation Interactions channel

Definition in file [SoxChannel.C](#).

## 11.400 SoxChannel.H File Reference

```
#include "Channels.H"
```

```
#include <vector>
```

Include dependency graph for SoxChannel.H:



This graph shows which files directly or indirectly include this file:



### Classes

- class [SoxChannel](#)  
*A short-range orientation interactions channel.*
- struct **SoxChannel::CacheElem**

#### 11.400.1 Detailed Description

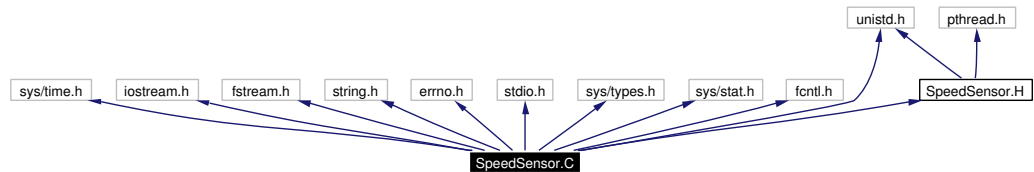
Shortrange Orientation Interactions channel

Definition in file [SoxChannel.H](#).

## 11.401 SpeedSensor.C File Reference

```
#include <sys/time.h>
#include <iostream.h>
#include <fstream.h>
#include <string.h>
#include <errno.h>
#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
#include "SpeedSensor.H"
```

Include dependency graph for SpeedSensor.C:



### Defines

- #define **SPEED\_CONVERSION** 1000
- #define **CHECK\_PERIOD** 10

### Functions

- void \* **SpeedSensor\_run** (void \*c0)

#### 11.401.1 Detailed Description

A class to interface with a mouse used as speed sensor

Definition in file [SpeedSensor.C](#).

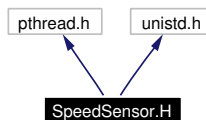


## 11.402 SpeedSensor.H File Reference

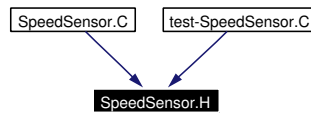
```
#include <pthread.h>
```

```
#include <unistd.h>
```

Include dependency graph for SpeedSensor.H:



This graph shows which files directly or indirectly include this file:



### Classes

- class [SpeedSensor](#)

*This class allows to use a PS/2 mouse as a speed sensor.*

#### 11.402.1 Detailed Description

A class to interface with a mouse used as speed sensor

Definition in file [SpeedSensor.H](#).

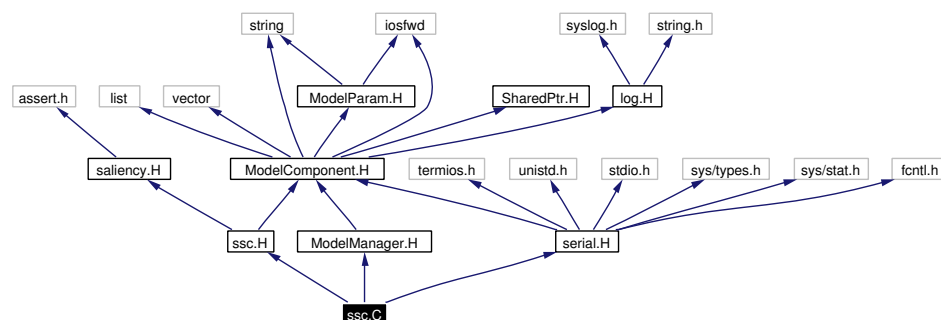
## 11.403 ssc.C File Reference

```
#include "ssc.H"
```

```
#include "ModelManager.H"
```

```
#include "serial.H"
```

Include dependency graph for ssc.C:

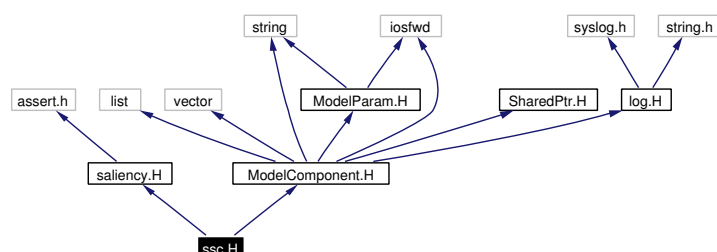


### 11.403.1 Detailed Description

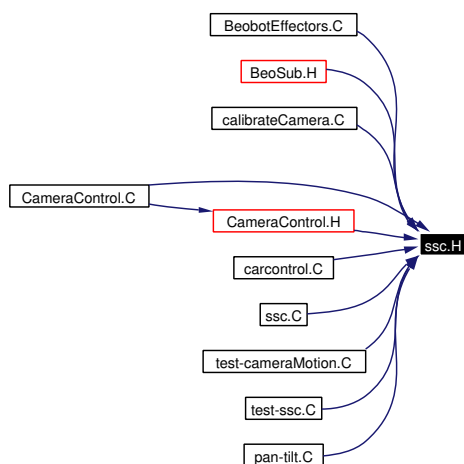
Interface to a [Serial](#) Servo Controller

Definition in file [ssc.C](#).

```
#include "saliency.H"
#include "ModelComponent.H"
Include dependency graph for ssc.H:
```



This graph shows which files directly or indirectly include this file:



## Classes

- class **SSC**

*Interface to a serial-servo-controller.*

## Defines

- `#define SSCNUMSERVOS 8`

### 11.404.1 Detailed Description

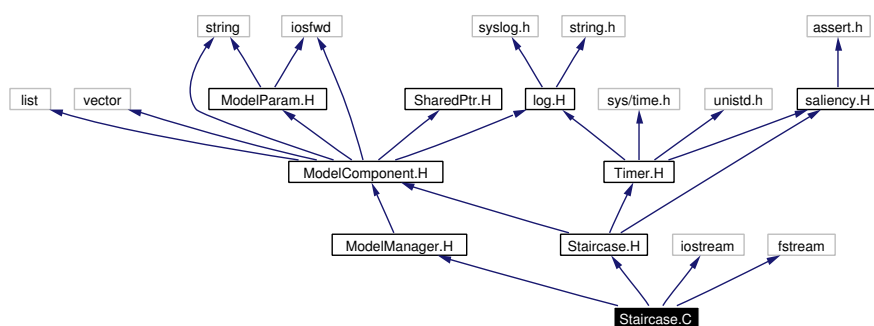
Interface to a serial-servo-controller

Definition in file [ssc.H](#).

## 11.405 Staircase.C File Reference

```
#include "Staircase.H"
#include "ModelManager.H"
#include <iostream>
#include <fstream>
```

Include dependency graph for Staircase.C:



### 11.405.1 Detailed Description

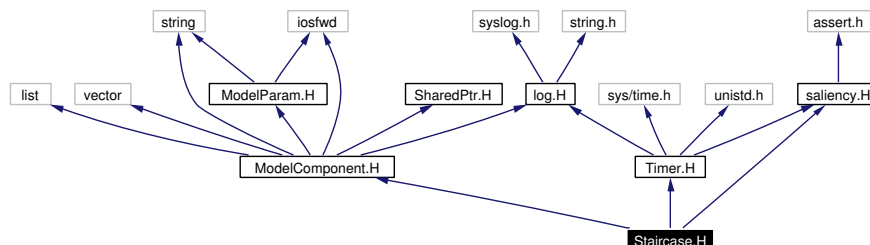
A staircase procedure for psychophysical thresholds

Definition in file [Staircase.C](#).

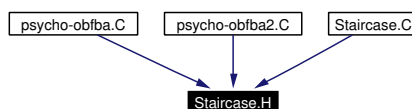
## 11.406 Staircase.H File Reference

```
#include "saliency.H"
#include "ModelComponent.H"
#include "Timer.H"
```

Include dependency graph for Staircase.H:



This graph shows which files directly or indirectly include this file:



### Classes

- class [Staircase](#)  
*A staircase procedure to measure psychophysical thresholds.*
- struct **Staircase::StaircaseEvent**

### 11.406.1 Detailed Description

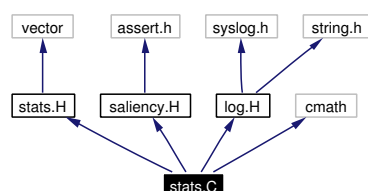
A staircase procedure for psychophysical thresholds

Definition in file [Staircase.H](#).

## 11.407 stats.C File Reference

```
#include "stats.H"  
#include "saliency.H"  
#include "log.H"  
#include <cmath>
```

Include dependency graph for stats.C:



This graph shows which files directly or indirectly include this file:



### 11.407.1 Detailed Description

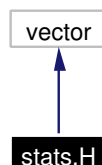
STATS classes

Definition in file [stats.C](#).

## 11.408 stats.H File Reference

```
#include <vector>
```

Include dependency graph for stats.H:



This graph shows which files directly or indirectly include this file:

### Classes

- class **stats**

### 11.408.1 Detailed Description

STATS classes

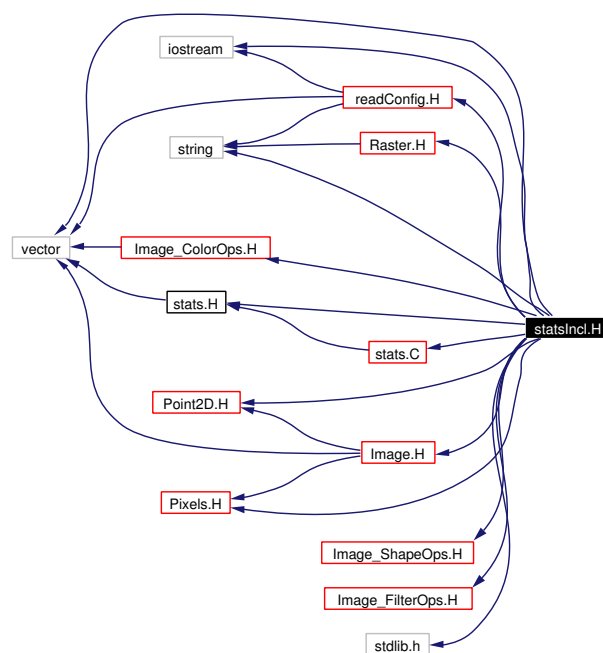
Definition in file [stats.H](#).



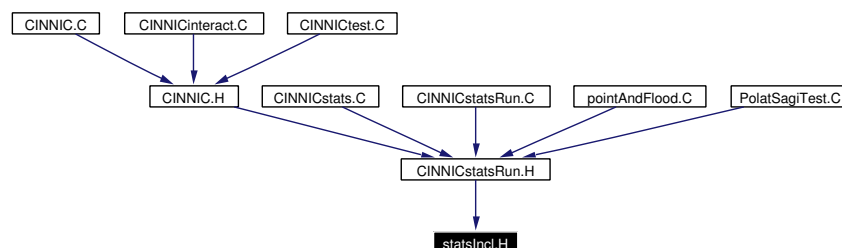
## 11.409 statsIncl.H File Reference

```
#include "readConfig.H"  
#include "Image.H"  
#include "Image_ColorOps.H"  
#include "Image_ShapeOps.H"  
#include "Image_FilterOps.H"  
#include "stats.H"  
#include "stats.C"  
#include "Raster.H"  
#include "Pixels.H"  
#include "Point2D.H"  
#include <stdlib.h>  
#include <iostream>  
#include <string>  
#include <vector>
```

Include dependency graph for statsIncl.H:



This graph shows which files directly or indirectly include this file:



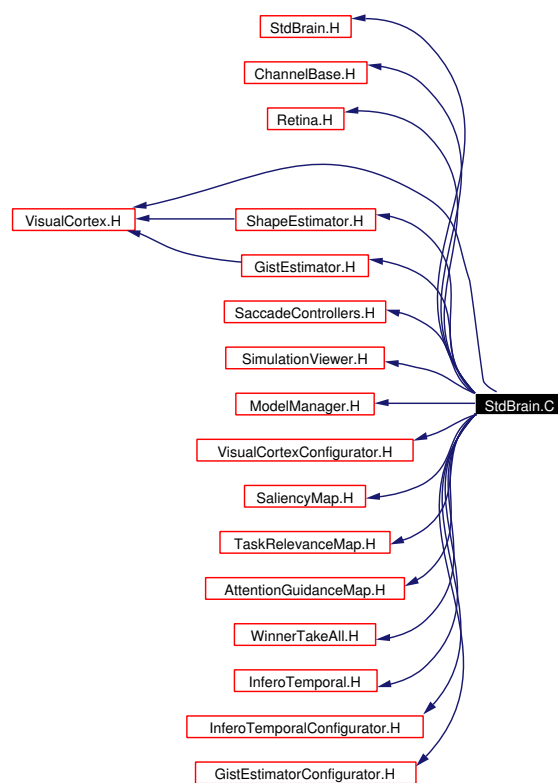
### 11.409.1 Detailed Description

FIXME

Definition in file [statsIncl.H](#).

## 11.410 StdBrain.C File Reference

```
#include "StdBrain.H"
#include "ChannelBase.H"
#include "Retina.H"
#include "VisualCortex.H"
#include "SaccadeControllers.H"
#include "SimulationViewer.H"
#include "ShapeEstimator.H"
#include "ModelManager.H"
#include "VisualCortexConfigurator.H"
#include "SaliencyMap.H"
#include "TaskRelevanceMap.H"
#include "AttentionGuidanceMap.H"
#include "WinnerTakeAll.H"
#include "InferoTemporal.H"
#include "InferoTemporalConfigurator.H"
#include "GistEstimator.H"
#include "GistEstimatorConfigurator.H"
Include dependency graph for StdBrain.C:
```



### 11.410.1 Detailed Description

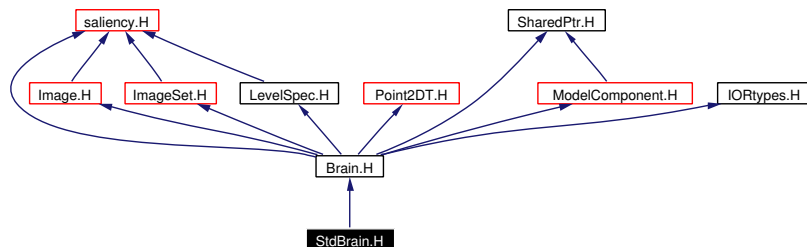
A canonical brain, which a standard set of channels.

Definition in file [StdBrain.C](#).

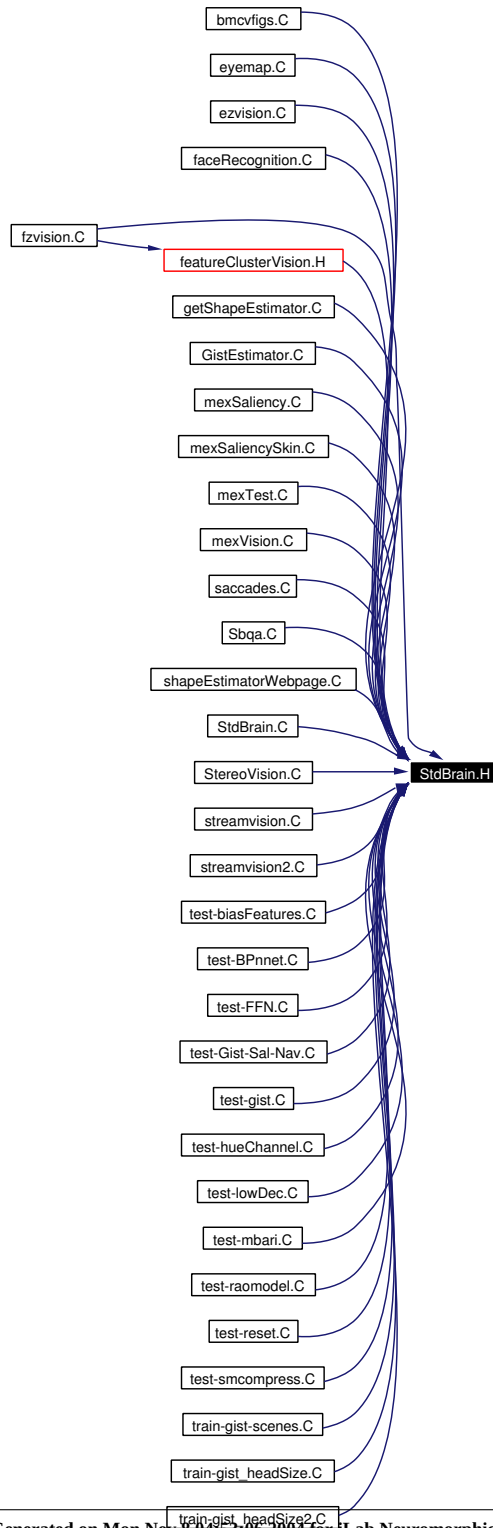
## 11.411 StdBrain.H File Reference

```
#include "Brain.H"
```

Include dependency graph for StdBrain.H:



This graph shows which files directly or indirectly include this file:



## Classes

- class [StdBrain](#)

*A subclass of [Brain](#) that automatically sets up the standard modules.*

### 11.411.1 Detailed Description

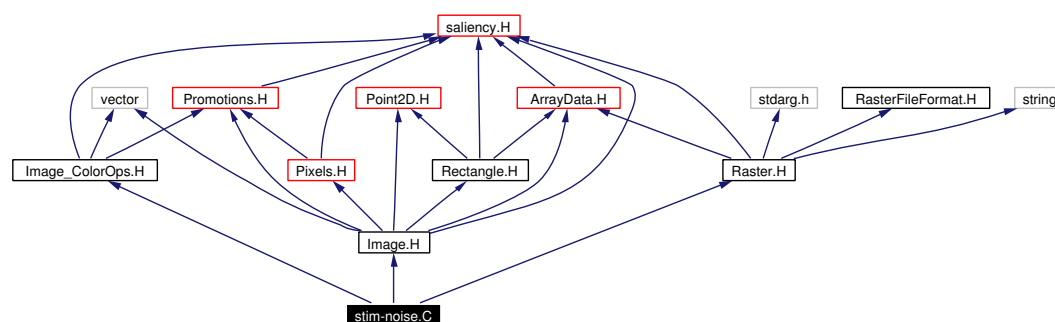
A canonical brain, which a standard set of channels.

Definition in file [StdBrain.H](#).

## 11.412 stim-noise.C File Reference

```
#include "Image.H"  
#include "Raster.H"  
#include "Image_ColorOps.H"
```

Include dependency graph for stim-noise.C:



### Functions

- int **main** (const int argc, const char \*\*argv)

#### 11.412.1 Detailed Description

generate noise stimuli

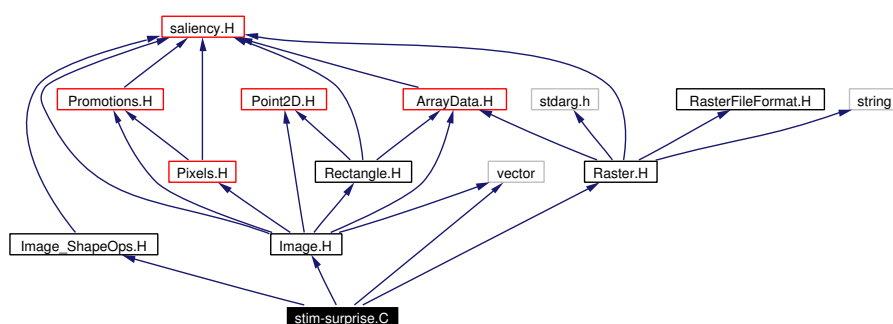
Definition in file [stim-noise.C](#).



## 11.413 stim-surprise.C File Reference

```
#include "Image.H"
#include "Raster.H"
#include "Image_ShapeOps.H"
#include <vector>
```

Include dependency graph for stim-surprise.C:



### Defines

- `#define WIDTH 256`
- `#define NFRAME 300`
- `#define DX 32`
- `#define DTHETA 0.1`
- `#define RAD 8`

### Functions

- `int main (const int argc, const char **argv)`

#### 11.413.1 Detailed Description

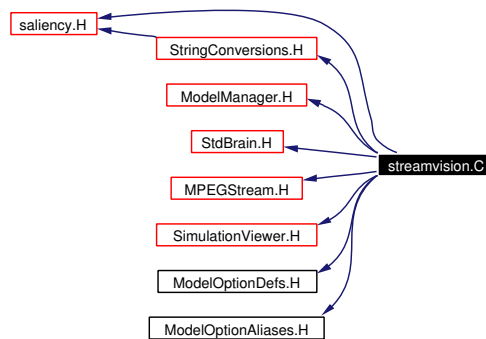
generate surprising stimuli

Definition in file [stim-surprise.C](#).

## 11.414 streamvision.C File Reference

```
#include "saliency.H"  
#include "ModelManager.H"  
#include "StdBrain.H"  
#include "MPEGStream.H"  
#include "StringConversions.H"  
#include "SimulationViewer.H"  
#include "ModelOptionDefs.H"  
#include "ModelOptionAliases.H"
```

Include dependency graph for streamvision.C:



## Functions

- `int main (const int argc, const char **argv)`

### 11.414.1 Detailed Description

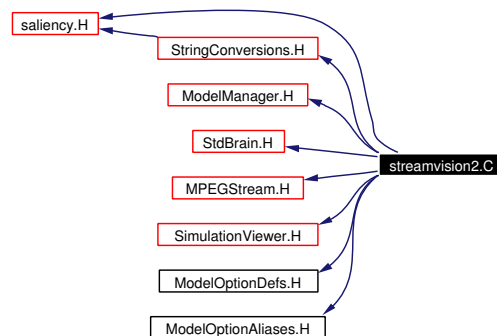
restricted version of [ezvision.C](#) except that it uses [MPEGStream](#) for frame I/O instead of [FrameSeries](#). For now, only attention trajectories may be saved as results.

Definition in file [streamvision.C](#).

## 11.415 streamvision2.C File Reference

```
#include "saliency.H"  
#include "ModelManager.H"  
#include "StdBrain.H"  
#include "MPEGStream.H"  
#include "StringConversions.H"  
#include "SimulationViewer.H"  
#include "ModelOptionDefs.H"  
#include "ModelOptionAliases.H"
```

Include dependency graph for streamvision2.C:



### Functions

- int [main](#) (const int argc, const char \*\*argv)

*Variable-framerate version of [streamvision.C](#).*

### 11.415.1 Detailed Description

restricted version of [ezvision.C](#) except that it uses [MPEGStream](#) for frame I/O instead of [FrameSeries](#). For now, only attention trajectories may be saved as results.

Definition in file [streamvision2.C](#).

## 11.415.2 Function Documentation

### 11.415.2.1 `int main (const int argc, const char ** argv)`

Variable-framerate version of [streamvision.C](#).

This version of streamvision is for variable framerate. It expects a .fl file, which should contain, for each line:

name time

where name is a string (no spaces) and time is the absolute time (in ms) at which the frame should be loaded and processed.

Definition at line 57 of file streamvision2.C.

References `decXY()`, `SharedPtr< T >::get()`, `Image< T >::getHeight()`, `Image< T >::getWidth()`, `lowPass3()`, `main()`, `MYLOGVERB`, `OPTEXP_ALL`, `OPTEXP_SAVE`, and `REQUEST_OPTIONALIAS_ATTENTION`.

## 11.416 StringConversions.C File Reference

```
#include "log.H"
#include "StringConversions.H"
#include <sstream>
#include "saliency.H"
#include "ArrayData.H"
#include "Point2D.H"
#include "FrameRange.H"
#include "LevelSpec.H"
#include "Rectangle.H"
#include "Pixels.H"
#include <netinet/in.h>
#include "InstantiateAll.H"
```

Include dependency graph for StringConversions.C:

### Defines

- #define **INstantiateAll**(T)

### Functions

- template void **convertToString** (const int &val, [std::string](#) &str)
- template bool **convertFromString** (const [std::string](#) &str, int &val)
- template void **convertToString** (const short int &val, [std::string](#) &str)
- template bool **convertFromString** (const [std::string](#) &str, short int &val)
- template void **convertToString** (const [uint](#) &val, [std::string](#) &str)
- template bool **convertFromString** (const [std::string](#) &str, [uint](#) &val)
- template void **convertToString** (const float &val, [std::string](#) &str)
- template bool **convertFromString** (const [std::string](#) &str, float &val)
- template void **convertToString** (const double &val, [std::string](#) &str)
- template bool **convertFromString** (const [std::string](#) &str, double &val)
- template<class T> void **convertToString** (const T &val, [std::string](#) &str)
- template<class T> bool **convertFromString** (const [std::string](#) &str, T &val)
- void **convertToString** (const [std::string](#) &val, [std::string](#) &str)

[std::string](#) specialization

- bool `convertFromString` (const `std::string` &str, `std::string` &val)  
*`std::string` specialization*
- void `convertToString` (const `Dims` &val, `std::string` &str)  
*`Dims` specialization: format is "<int>x<int>"*.
- bool `convertFromString` (const `std::string` &str, `Dims` &val)  
*`Dims` specialization: format is "<int>x<int>"*.
- void `convertToString` (const `Point2D` &val, `std::string` &str)  
*`Point2D` specialization: format is "<int>,<int>"*.
- bool `convertFromString` (const `std::string` &str, `Point2D` &val)  
*`Point2D` specialization: format is "<int>,<int>"*.
- void `convertToString` (const bool &val, `std::string` &str)  
*`bool` specialization: format is "true|false"*
- bool `convertFromString` (const `std::string` &str, bool &val)  
*`bool` specialization*
- void `convertToString` (const `FrameGrabberMode` val, `std::string` &str)  
*Conversion from `FrameGrabberMode` to string.*
- bool `convertFromString` (const `std::string` &str, `FrameGrabberMode` &val)  
*Conversion from string to `FrameGrabberMode`.*
- void `convertToString` (const `FrameRange` &val, `std::string` &str)  
*`FrameRange` specialization: format is "<int>-<int>@<float>"*.
- bool `convertFromString` (const `std::string` &str, `FrameRange` &val)  
*`FrameRange` specialization: format is "<int>-<int>@<float>"*.
- void `convertToString` (const `LevelSpec` &val, `std::string` &str)  
*`LevelSpec` specialization: format is "<int>-<int>,<int>-<int>,<int>"*.
- bool `convertFromString` (const `std::string` &str, `LevelSpec` &val)  
*`LevelSpec` specialization: format is "<int>-<int>,<int>-<int>,<int>"*.
- void `convertToString` (const `Rectangle` &val, `std::string` &str)  
*`Rectangle` specialization: format is "<int>,<int>,<int>,<int>"*.

- bool `convertFromString` (const `std::string` &str, `Rectangle` &val)  
*Rectangle specialization: format is "<int>,<int>,<int>,<int>".*
- void `convertToString` (const `RasterFileFormat` val, `std::string` &str)  
*RasterFileFormat specialization.*
- bool `convertFromString` (const `std::string` &str, `RasterFileFormat` &val)  
*RasterFileFormat specialization.*
- void `convertToString` (const `PixRGB< byte >` &val, `std::string` &str)  
*PixRGB<byte> specialization.*
- bool `convertFromString` (const `std::string` &str, `PixRGB< byte >` &val)  
*RasterFileFormat specialization.*
- void `convertToString` (const `VisualFeature` val, `std::string` &str)  
*VisualFeature specialization.*
- bool `convertFromString` (const `std::string` &str, `VisualFeature` &val)  
*VisualFeature specialization.*
- void `convertToString` (const `PyramidType` val, `std::string` &str)  
*PyramidType specialization.*
- bool `convertFromString` (const `std::string` &str, `PyramidType` &val)  
*PyramidType specialization.*
- void `convertToString` (const `MaxNormType` val, `std::string` &str)  
*MaxNormType specialization.*
- bool `convertFromString` (const `std::string` &str, `MaxNormType` &val)  
*MaxNormType specialization.*
- void `convertToString` (const `IORtype` val, `std::string` &str)  
*IORtype specialization.*
- bool `convertFromString` (const `std::string` &str, `IORtype` &val)  
*IORtype specialization.*
- void `convertToString` (const `ShapeEstimatorMode` val, `std::string` &str)  
*ShapeEstimatorMode specialization.*

- bool `convertFromString` (const `std::string` &str, `ShapeEstimatorMode` &val)  
*ShapeEstimatorMode specialization.*
- void `convertToString` (const `ShapeEstimatorSmoothMethod` val, `std::string` &str)  
*ShapeEstimatorSmoothType specialization.*
- bool `convertFromString` (const `std::string` &str, `ShapeEstimatorSmoothMethod` &val)  
*ShapeEstimatorSmoothType specialization.*
- void `convertToString` (const `OrientComputeType` val, `std::string` &str)  
*OrientComputeType specialization.*
- bool `convertFromString` (const `std::string` &str, `OrientComputeType` &val)  
*OrientComputeType specialization.*
- void `convertToString` (const in\_addr &val, `std::string` &str)  
*in\_addr specialization: format is "<int>.<int>.<int>.<int>"*
- bool `convertFromString` (const `std::string` &str, in\_addr &val)  
*in\_addr specialization: format is "<int>.<int>.<int>.<int>"*
- void `convertToString` (const `BitObjectDrawMode` val, `std::string` &str)  
*BitObjectDrawMode specialization.*
- bool `convertFromString` (const `std::string` &str, `BitObjectDrawMode` &val)  
*BitObjectDrawMode specialization.*
- void `convertToString` (const `ImageDisplayType` val, `std::string` &str)  
*ImageDisplayType specialization.*
- bool `convertFromString` (const `std::string` &str, `ImageDisplayType` &val)  
*ImageDisplayType specialization.*
- void `convertToString` (const `ColorComputeType` val, `std::string` &str)  
*ColorComputeType specialization.*
- bool `convertFromString` (const `std::string` &str, `ColorComputeType` &val)  
*ColorComputeType specialization.*
- template<class T> `std::string toStr` (const T &val)  
*converts val into a string and returns the string*



- `template<class T> T fromStr (const std::string &str)`  
*converts str into a T value and returns the T value*

### 11.416.1 Detailed Description

Convert to and from [std::string](#)

Definition in file [StringConversions.C](#).

### 11.416.2 Define Documentation

#### 11.416.2.1 `#define` INSTANTIATE(T)

**Value:**

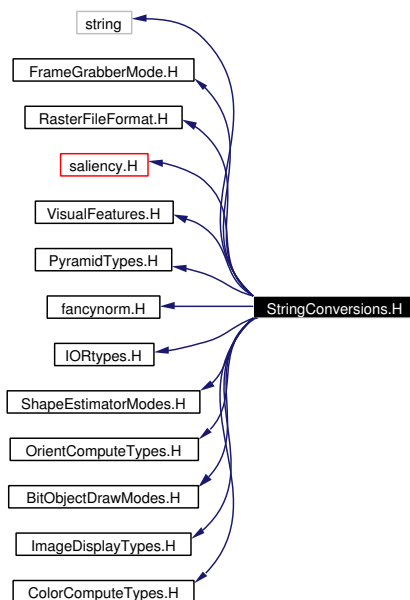
```
template std::string toStr(const T& val); \
    template T fromStr(const std::string& str) ;
```

Definition at line 63 of file StringConversions.C.

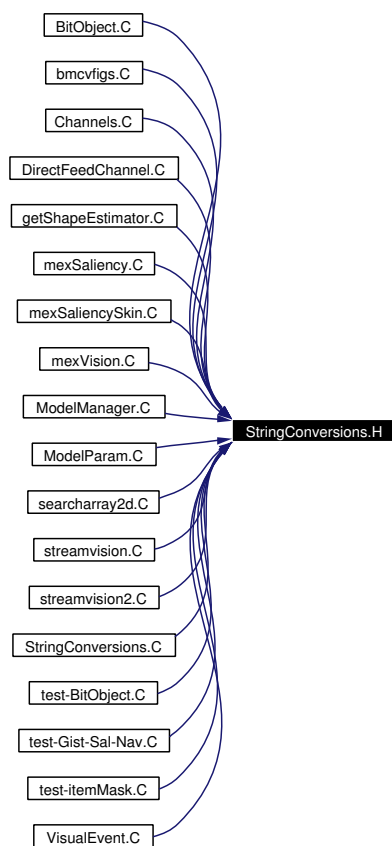
## 11.417 StringConversions.H File Reference

```
#include <string>
#include "FrameGrabberMode.H"
#include "RasterFileFormat.H"
#include "saliency.H"
#include "VisualFeatures.H"
#include "PyramidTypes.H"
#include "fancynorm.H"
#include "IORtypes.H"
#include "ShapeEstimatorModes.H"
#include "OrientComputeTypes.H"
#include "BitObjectDrawModes.H"
#include "ImageDisplayTypes.H"
#include "ColorComputeTypes.H"
```

Include dependency graph for StringConversions.H:



This graph shows which files directly or indirectly include this file:



## Generic string conversion functions

- `template<class T> void convertToString (const T &val, std::string &str)`
- `template<class T> bool convertFromString (const std::string &str, T &val)`

## Specialized implementations of string conversion functions

- `void convertToString (const std::string &val, std::string &str)`  
*`std::string` specialization*
- `bool convertFromString (const std::string &str, std::string &val)`  
*`std::string` specialization*
- `void convertToString (const Dims &val, std::string &str)`  
*`Dims` specialization: format is "<int>x<int>"*

- bool `convertFromString` (const `std::string` &str, `Dims` &val)  
*Dims specialization: format is "<int>x<int>".*
- void `convertToString` (const `Point2D` &val, `std::string` &str)  
*Point2D specialization: format is "<int>,<int>".*
- bool `convertFromString` (const `std::string` &str, `Point2D` &val)  
*Point2D specialization: format is "<int>,<int>".*
- void `convertToString` (const bool &val, `std::string` &str)  
*bool specialization: format is "true|false"*
- bool `convertFromString` (const `std::string` &str, bool &val)  
*bool specialization*
- void `convertToString` (const `FrameGrabberMode` val, `std::string` &str)  
*Conversion from FrameGrabberMode to string.*
- bool `convertFromString` (const `std::string` &str, `FrameGrabberMode` &val)  
*Conversion from string to FrameGrabberMode.*
- void `convertToString` (const `FrameRange` &val, `std::string` &str)  
*FrameRange specialization: format is "<int>-<int>@<float>".*
- bool `convertFromString` (const `std::string` &str, `FrameRange` &val)  
*FrameRange specialization: format is "<int>-<int>@<float>".*
- void `convertToString` (const `LevelSpec` &val, `std::string` &str)  
*LevelSpec specialization: format is "<int>-<int>,<int>-<int>,<int>".*
- bool `convertFromString` (const `std::string` &str, `LevelSpec` &val)  
*LevelSpec specialization: format is "<int>-<int>,<int>-<int>,<int>".*
- void `convertToString` (const `Rectangle` &val, `std::string` &str)  
*Rectangle specialization: format is "<int>,<int>,<int>,<int>".*
- bool `convertFromString` (const `std::string` &str, `Rectangle` &val)  
*Rectangle specialization: format is "<int>,<int>,<int>,<int>".*
- void `convertToString` (const `RasterFileFormat` val, `std::string` &str)  
*RasterFileFormat specialization.*

- bool `convertFromString` (const `std::string` &str, `RasterFileFormat` &val)  
*RasterFileFormat specialization.*
- void `convertToString` (const `PixRGB< byte >` &val, `std::string` &str)  
*PixRGB<byte> specialization.*
- bool `convertFromString` (const `std::string` &str, `PixRGB< byte >` &val)  
*RasterFileFormat specialization.*
- void `convertToString` (const `VisualFeature` val, `std::string` &str)  
*VisualFeature specialization.*
- bool `convertFromString` (const `std::string` &str, `VisualFeature` &val)  
*VisualFeature specialization.*
- void `convertToString` (const `PyramidType` val, `std::string` &str)  
*PyramidType specialization.*
- bool `convertFromString` (const `std::string` &str, `PyramidType` &val)  
*PyramidType specialization.*
- void `convertToString` (const `MaxNormType` val, `std::string` &str)  
*MaxNormType specialization.*
- bool `convertFromString` (const `std::string` &str, `MaxNormType` &val)  
*MaxNormType specialization.*
- void `convertToString` (const `IORtype` val, `std::string` &str)  
*IORtype specialization.*
- bool `convertFromString` (const `std::string` &str, `IORtype` &val)  
*IORtype specialization.*
- void `convertToString` (const `ShapeEstimatorMode` val, `std::string` &str)  
*ShapeEstimatorMode specialization.*
- bool `convertFromString` (const `std::string` &str, `ShapeEstimatorMode` &val)  
*ShapeEstimatorMode specialization.*
- void `convertToString` (const `ShapeEstimatorSmoothMethod` val, `std::string` &str)

*ShapeEstimatorSmoothType specialization.*

- bool `convertFromString` (const `std::string` &str, `ShapeEstimatorSmoothMethod` &val)

*ShapeEstimatorSmoothType specialization.*

- void `convertToString` (const `OrientComputeType` val, `std::string` &str)

*OrientComputeType specialization.*

- bool `convertFromString` (const `std::string` &str, `OrientComputeType` &val)

*OrientComputeType specialization.*

- void `convertToString` (const in\_addr &val, `std::string` &str)

*in\_addr specialization: format is "<int>.<int>.<int>.<int>"*

- bool `convertFromString` (const `std::string` &str, in\_addr &val)

*in\_addr specialization: format is "<int>.<int>.<int>.<int>"*

- void `convertToString` (const `BitObjectDrawMode` val, `std::string` &str)

*BitObjectDrawMode specialization.*

- bool `convertFromString` (const `std::string` &str, `BitObjectDrawMode` &val)

*BitObjectDrawMode specialization.*

- void `convertToString` (const `ImageDisplayType` val, `std::string` &str)

*ImageDisplayType specialization.*

- bool `convertFromString` (const `std::string` &str, `ImageDisplayType` &val)

*ImageDisplayType specialization.*

- void `convertToString` (const `ColorComputeType` val, `std::string` &str)

*ColorComputeType specialization.*

- bool `convertFromString` (const `std::string` &str, `ColorComputeType` &val)

*ColorComputeType specialization.*

## Easier-syntax functions to convert to/from string

- template<class T> `std::string toStr` (const T &val)

*converts val into a string and returns the string*

- `template<class T> T fromStr (const std::string &str)`  
*converts str into a T value and returns the T value*

### 11.417.1 Detailed Description

Convert to and from [std::string](#)

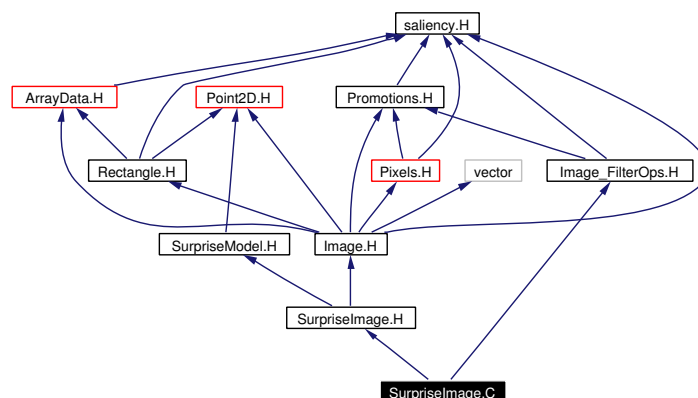
Definition in file [StringConversions.H](#).

## 11.418 SurpriseImage.C File Reference

```
#include "SurpriseImage.H"
```

```
#include "Image_FilterOps.H"
```

Include dependency graph for SurpriseImage.C:



### 11.418.1 Detailed Description

a 2D array of [SurpriseModel](#) objects

Definition in file [SurpriseImage.C](#).

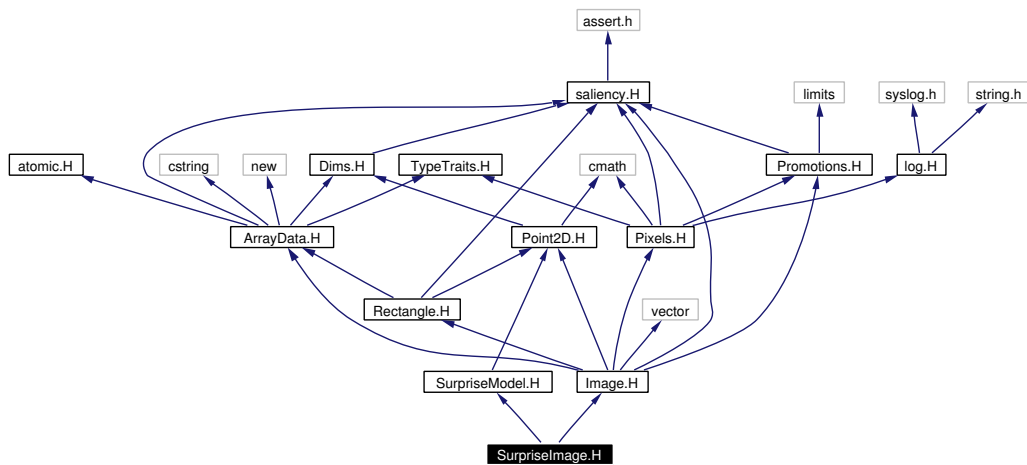


## 11.419 SurpriseImage.H File Reference

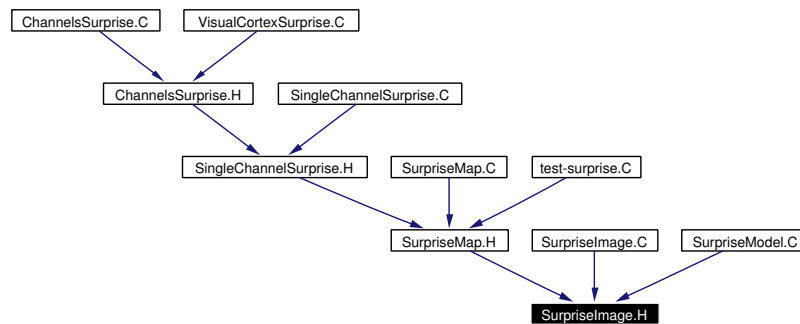
```
#include "Image.H"
```

```
#include "SurpriseModel.H"
```

Include dependency graph for SurpriseImage.H:



This graph shows which files directly or indirectly include this file:



## Classes

- class [SurpriseImage](#)  
A 2D array of [SurpriseModel](#) objects.

### 11.419.1 Detailed Description

a 2D array of [SurpriseModel](#) objects

Definition in file [SurpriseImage.H](#).

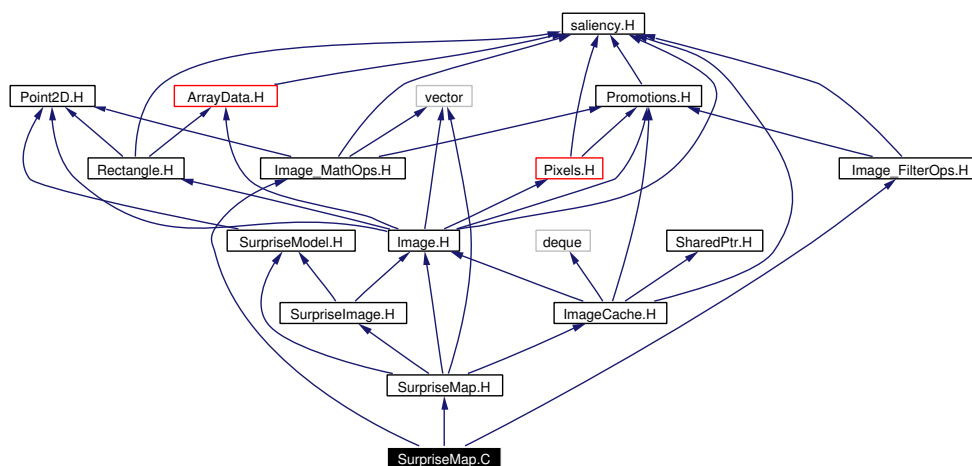
## 11.420 SurpriseMap.C File Reference

```
#include "SurpriseMap.H"
```

```
#include "Image_FilterOps.H"
```

```
#include "Image_MathOps.H"
```

Include dependency graph for SurpriseMap.C:



### 11.420.1 Detailed Description

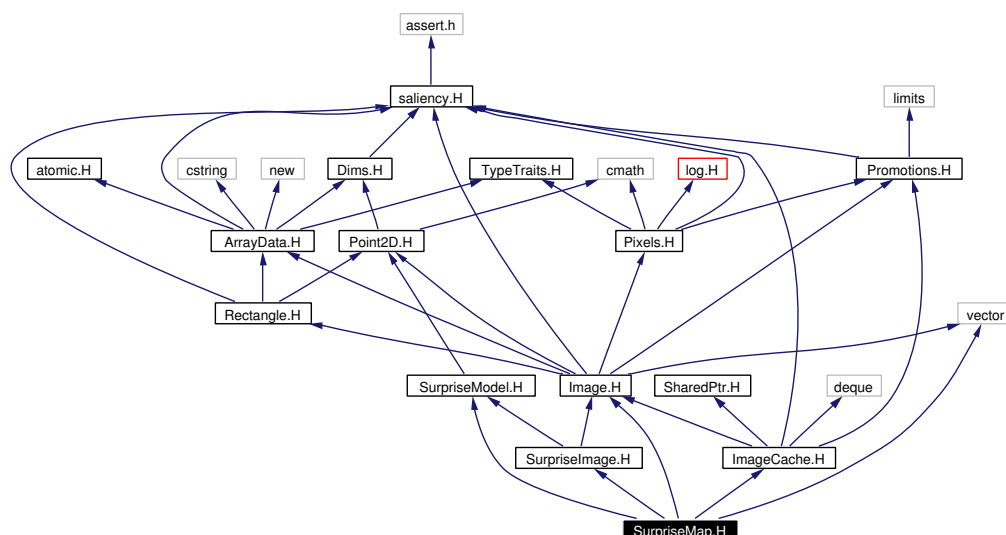
a surprise map

Definition in file [SurpriseMap.C](#).

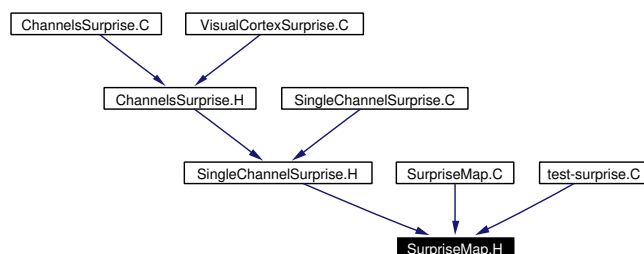
## 11.421 SurpriseMap.H File Reference

```
#include "Image.H"
#include "SurpriseModel.H"
#include "SurpriseImage.H"
#include "ImageCache.H"
#include <vector>
```

Include dependency graph for SurpriseMap.H:



This graph shows which files directly or indirectly include this file:



## Classes

- class [SurpriseMap](#)  
*A Surprise [Map](#).*

### 11.421.1 Detailed Description

a surprise map

Definition in file [SurpriseMap.H](#).

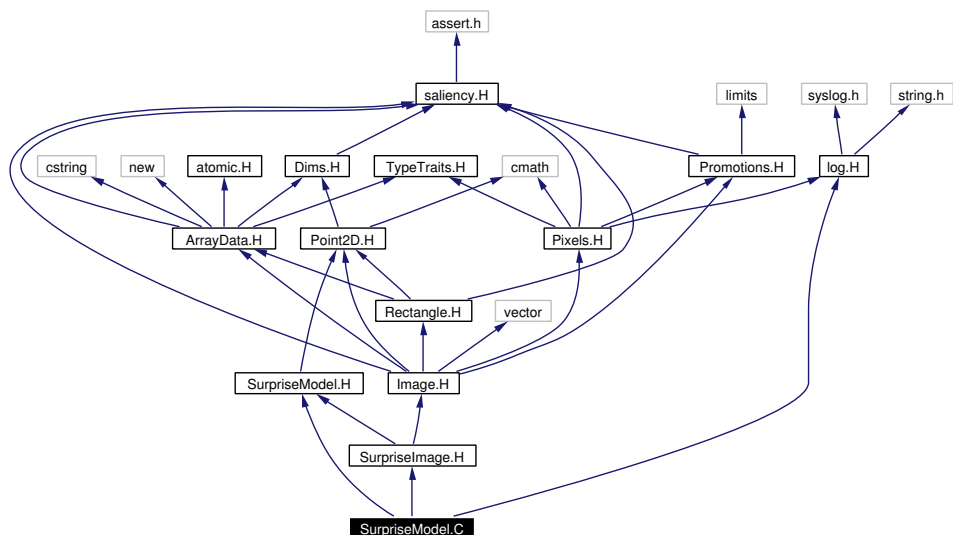
## 11.422 SurpriseModel.C File Reference

```
#include "SurpriseModel.H"
```

```
#include "SurpriseImage.H"
```

```
#include "log.H"
```

Include dependency graph for SurpriseModel.C:



### Classes

- struct **cheb\_series\_struct**

### Typedefs

- typedef cheb\_series\_struct **cheb\_series**

### Functions

- double **cheb\_eval** (const cheb\_series \*cs, const double x)
- double **psi** (const double x)

*Compute the psi (derivative of log of gamma) function (requires  $x > 0.0$ ).*

## Variables

- double **psics\_data** [23]
- cheb\_series **psi\_cs** = { psics\_data, 22, -1, 1, 17 }
- double **apsics\_data** [16]
- cheb\_series **apsi\_cs** = { apsics\_data, 15, -1, 1, 9 }

### 11.422.1 Detailed Description

a local (single-point) model of surprise

Definition in file [SurpriseModel.C](#).

### 11.422.2 Function Documentation

#### 11.422.2.1 double psi (const double x)

Compute the psi (derivative of log of gamma) function (requires  $x > 0.0$ ).

Definition at line 455 of file SurpriseModel.C.

Referenced by gabor2(), gaborFilter(), gaborFilter2(), gaborFilter3(), grating(), and SurpriseModelSP::surprise().

### 11.422.3 Variable Documentation

#### 11.422.3.1 double apsics\_data[16] [static]

Initial value:

```
{
- .0204749044678185,
- .0101801271534859,
.0000559718725387,
- .0000012917176570,
.0000000572858606,
- .0000000038213539,
.0000000003397434,
- .0000000000374838,
.0000000000048990,
- .0000000000007344,
.0000000000001233,
- .0000000000000228,
.0000000000000045,
- .0000000000000009,
.0000000000000002,
- .0000000000000000
}
```

Definition at line 413 of file SurpriseModel.C.

### 11.422.3.2 double psics\_data[23] [static]

**Initial value:**

```
{  
- .038057080835217922,  
 .491415393029387130,  
- .056815747821244730,  
 .008357821225914313,  
- .001333232857994342,  
 .000220313287069308,  
- .000037040238178456,  
 .000006283793654854,  
- .000001071263908506,  
 .000000183128394654,  
- .000000031353509361,  
 .000000005372808776,  
- .000000000921168141,  
 .000000000157981265,  
- .00000000027098646,  
 .00000000004648722,  
- .00000000000797527,  
 .00000000000136827,  
- .00000000000023475,  
 .0000000000004027,  
- .00000000000000691,  
 .00000000000000118,  
- .00000000000000020  
}
```

Definition at line 386 of file SurpriseModel.C.

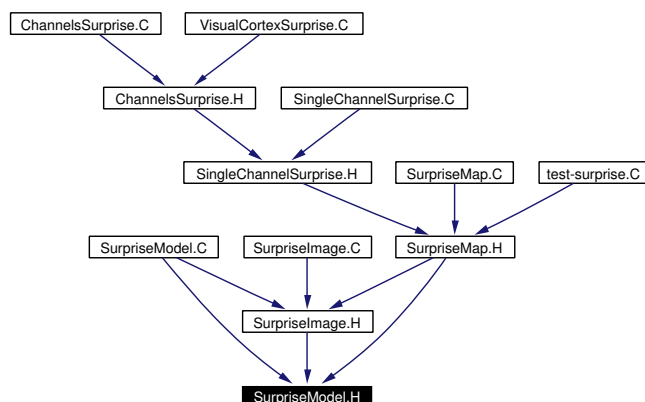


## 11.423 SurpriseModel.H File Reference

```
#include "Point2D.H"
```

Include dependency graph for SurpriseModel.H:

This graph shows which files directly or indirectly include this file:



### Classes

- class [SurpriseModel](#)  
*A local (single-point) surprise model.*
- class [SurpriseModelSG](#)  
*A single-Gaussian [SurpriseModel](#).*
- class [SurpriseModelSP](#)  
*A single-Poisson/Gamma [SurpriseModel](#).*

### Functions

- double [psi](#) (const double x)  
*Compute the psi (derivative of log of gamma) function (requires  $x > 0.0$ ).*

#### 11.423.1 Detailed Description

a local (single-point) model of surprise

Definition in file [SurpriseModel.H](#).

## 11.423.2 Function Documentation

### 11.423.2.1 `double psi (const double x)`

Compute the psi (derivative of log of gamma) function (requires  $x > 0.0$ ).

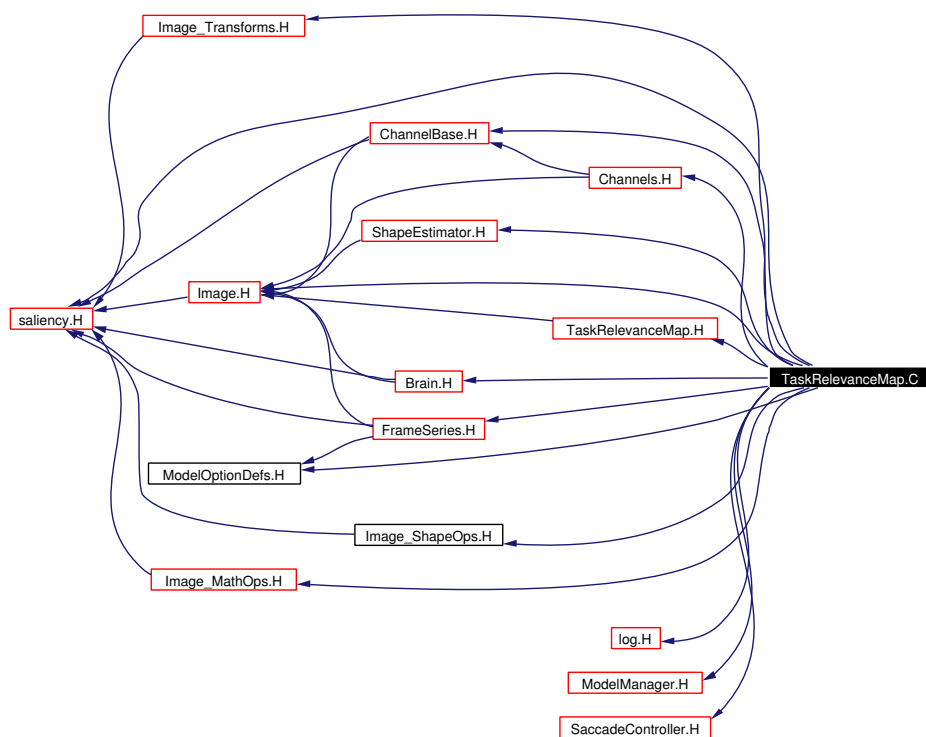
Definition at line 455 of file SurpriseModel.C.

Referenced by `gabor2()`, `gaborFilter()`, `gaborFilter2()`, `gaborFilter3()`, `grating()`, and `SurpriseModelSP::surprise()`.

## 11.424 TaskRelevanceMap.C File Reference

```
#include "saliency.H"
#include "Image.H"
#include "Image_ShapeOps.H"
#include "Image_MathOps.H"
#include "TaskRelevanceMap.H"
#include "log.H"
#include "ChannelBase.H"
#include "ModelOptionDefs.H"
#include "ModelManager.H"
#include "FrameSeries.H"
#include "ShapeEstimator.H"
#include "SaccadeController.H"
#include "Channels.H"
#include "Brain.H"
#include "Image_Transforms.H"
```

Include dependency graph for TaskRelevanceMap.C:



### 11.424.1 Detailed Description

Implementation for task-relevance map class

Definition in file [TaskRelevanceMap.C](#).

## 11.425 TaskRelevanceMap.H File Reference

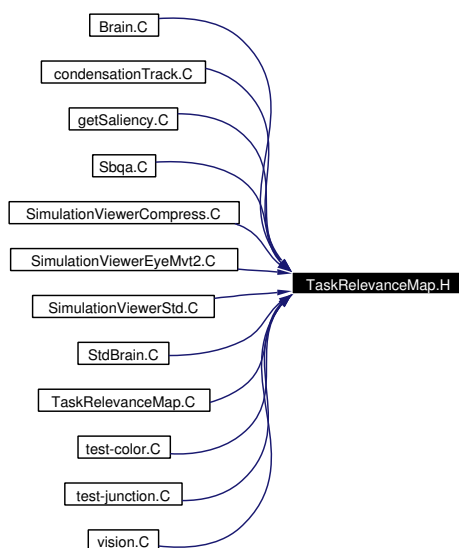
```
#include "Image.H"
```

```
#include "ModelComponent.H"
```

```
#include "ImageCache.H"
```

Include dependency graph for TaskRelevanceMap.H:

This graph shows which files directly or indirectly include this file:



### Classes

- class [TaskRelevanceMap](#)  
*The task-relevance map base class.*
- class [TaskRelevanceMapConfigurator](#)  
*TaskRelevanceMap configurator.*
- class [TaskRelevanceMapStd](#)  
*The standard task-relevance map.*
- class [TaskRelevanceMapKillStatic](#)  
*A killStatic task-relevance map.*
- class [TaskRelevanceMapKillN](#)

*A KillN task-relevance map.*

### 11.425.1 Detailed Description

Class declarations for task-relevance map class

Definition in file [TaskRelevanceMap.H](#).

## 11.426 TCPcliServ.C File Reference

```
#include "TCPdefs.H"
#include "TCPcliServ.H"
#include "log.H"
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <netdb.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <fcntl.h>
#include <errno.h>
#include <netinet/tcp.h>
#include <sys/shm.h>
#include <net/ethernet.h>
```

Include dependency graph for TCPcliServ.C:

### Defines

- #define [MYLOGID](#) fd  
*for ID-logging. See [log.H](#):*
- #define [TCPCS\\_UNKNOWN](#) 0  
*internal states:*
- #define [TCPCS\\_READING](#) 1
- #define [TCPCS\\_WRITING](#) 2
- #define [TCPCS\\_BOGUS](#) 4

### 11.426.1 Detailed Description

A client/server to receive/send [TCPmessage](#)

Definition in file [TCPcliServ.C](#).

## 11.426.2 Define Documentation

### 11.426.2.1 `#define MYLOGID fd`

for ID-logging. See [log.H](#):

Definition at line 55 of file TCPcliServ.C.

### 11.426.2.2 `#define TCPCS_UNKNOWN 0`

internal states:

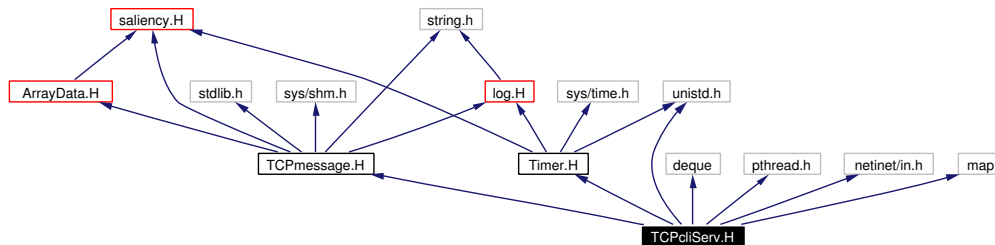
Definition at line 58 of file TCPcliServ.C.



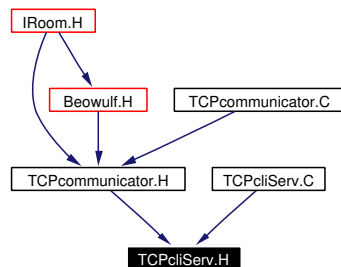
## 11.427 TCPcliServ.H File Reference

```
#include "TCPmessage.H"
#include <unistd.h>
#include <deque>
#include "Timer.H"
#include <pthread.h>
#include <netinet/in.h>
#include <map>
```

Include dependency graph for TCPcliServ.H:



This graph shows which files directly or indirectly include this file:



## Classes

- class [TCPcliServ](#)

*A client/server to receive/send [TCPmessage](#).*

### 11.427.1 Detailed Description

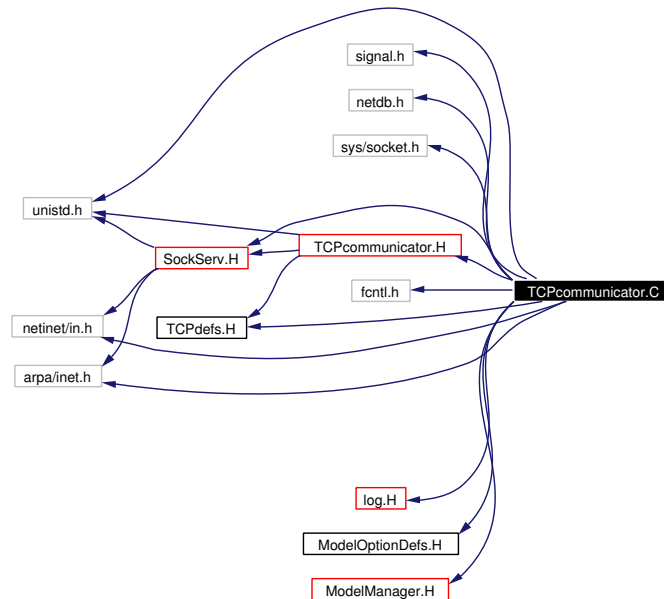
A client/server to receive/send [TCPmessage](#)

Definition in file [TCPcliServ.H](#).

## 11.428 TCPcommunicator.C File Reference

```
#include <unistd.h>
#include <signal.h>
#include <netdb.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <fcntl.h>
#include "TCPdefs.H"
#include "TCPcommunicator.H"
#include "SockServ.H"
#include "log.H"
#include "ModelOptionDefs.H"
#include "ModelManager.H"
```

Include dependency graph for TCPcommunicator.C:



## Defines

- `#define MYLOGID fd`  
*for id-logging; see [log.H](#):*

## Functions

- `void * TCPcommunicator_run (void *c0)`

## Variables

- `in_addr defaultIP = { 0 }`

### 11.428.1 Detailed Description

A class to handle multiple [TCPmessage](#) communications

Definition in file [TCPcommunicator.C](#).

### 11.428.2 Define Documentation

#### 11.428.2.1 `#define MYLOGID fd`

for id-logging; see [log.H](#):

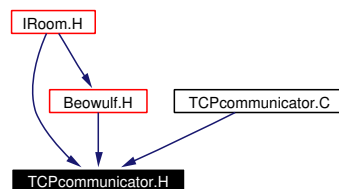
Definition at line 57 of file [TCPcommunicator.C](#).

## 11.429 TCPcommunicator.H File Reference

```
#include <unistd.h>
#include "TCPdefs.H"
#include "SockServ.H"
#include "TCPcliServ.H"
#include "ModelComponent.H"
#include <pthread.h>
#include <sys/time.h>
#include <sys/types.h>
```

Include dependency graph for TCPcommunicator.H:

This graph shows which files directly or indirectly include this file:



### Classes

- class [TCPcommunicator](#)  
*This class handles passing TCPmessages in and out.*

### Defines

- #define [POLL\\_SLEEP](#) 100  
*delay in us of sleep while polling for running thread:*

#### 11.429.1 Detailed Description

A class to handle multiple [TCPmessage](#) communications

Definition in file [TCPcommunicator.H](#).

## 11.429.2 Define Documentation

### 11.429.2.1 `#define POLL_SLEEP 100`

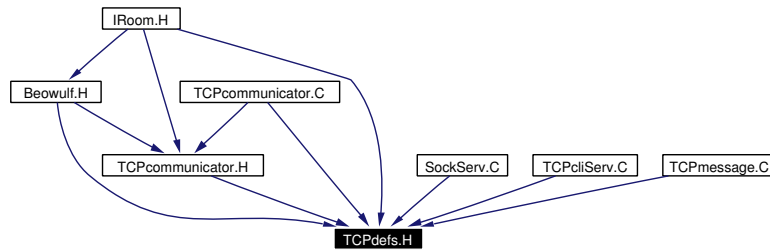
delay in us of sleep while polling for running thread:

Definition at line 52 of file TCPcommunicator.H.

Referenced by TCPcommunicator::pollvar().

## 11.430 TCPdefs.H File Reference

This graph shows which files directly or indirectly include this file:



### Defines

- #define **DONE** 0  
*return values for the various check() functions:*
- #define **BUG** 1
- #define **WAITREAD** 2
- #define **WAITWRITE** 3
- #define **WAITRW** 4
- #define **FINISH** 5
- #define **DEFPORT** 9789  
*default port to monitor:*
- #define **TCP\_QLEN** 20  
*TCPcommunicator config:.*
- #define **TCP\_TIMEOUT\_SEC** 0
- #define **TCP\_TIMEOUT\_USEC** 50

### 11.430.1 Detailed Description

Common definitions for the various TCPxxx classes

Definition in file [TCPdefs.H](#).

## 11.430.2 Define Documentation

### 11.430.2.1 `#define DEFPORT 9789`

default port to monitor:

Definition at line 49 of file TCPdefs.H.

Referenced by TCPcommunicator::contact().

### 11.430.2.2 `#define DONE 0`

return values for the various check() functions:

Definition at line 41 of file TCPdefs.H.

Referenced by TCPcliServ::check(), and TCPcommunicator::run().

### 11.430.2.3 `#define TCP_QLEN 20`

[TCPcommunicator](#) config:.

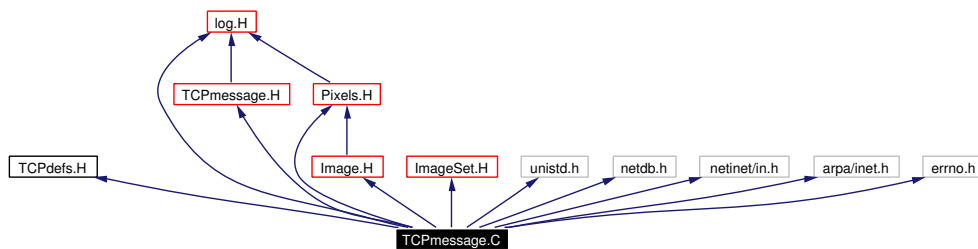
Definition at line 52 of file TCPdefs.H.



## 11.431 TCPmessage.C File Reference

```
#include "TCPdefs.H"
#include "TCPmessage.H"
#include "log.H"
#include "Image.H"
#include "Pixels.H"
#include "ImageSet.H"
#include <unistd.h>
#include <netdb.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <errno.h>
```

Include dependency graph for TCPmessage.C:



### Defines

- #define [MYLOGID](#) itsHead.itsId  
use message ID for ID-logging; see [log.H](#):

### 11.431.1 Detailed Description

Direct message passing over TCP connections

Definition in file [TCPmessage.C](#).

## 11.431.2 Define Documentation

### 11.431.2.1 `#define MYLOGID itsHead.itsId`

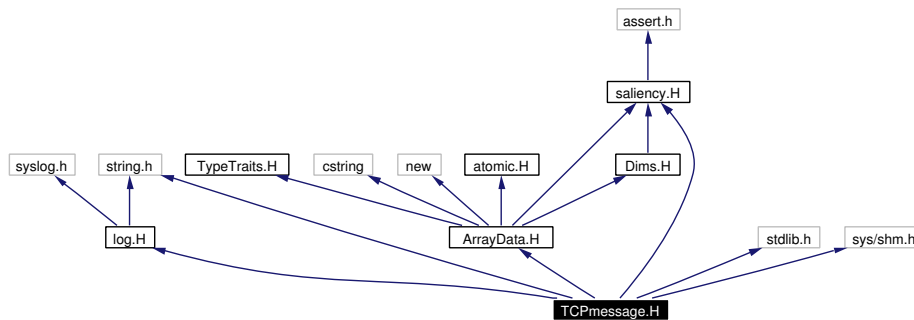
use message ID for ID-logging; see [log.H](#):

Definition at line 51 of file TCPmessage.C.

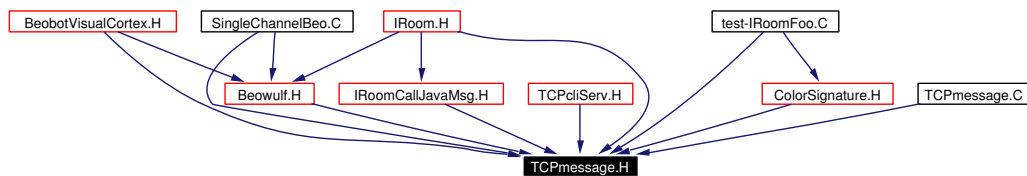
## 11.432 TCPmessage.H File Reference

```
#include "log.H"
#include "saliency.H"
#include "ArrayData.H"
#include <string.h>
#include <stdlib.h>
#include <sys/shm.h>
```

Include dependency graph for TCPmessage.H:



This graph shows which files directly or indirectly include this file:



## Classes

- class [TCPmessage](#)

*This class defines a modular message to be passed over TCP connections.*

- struct **TCPmessage::TCPmessageHeader**

*This private struct is the common header in all [TCPmessage](#) objects.*

## Defines

- `#define TCPMSG_END 0`  
*types of possible message fields:*
  - `#define TCPMSG_SHMINFO 1`
  - `#define TCPMSG_SHMDONE 2`
  - `#define TCPMSG_COLBYTEIMA 100`
  - `#define TCPMSG_FIXATION 101`
  - `#define TCPMSG_STRING 102`
  - `#define TCPMSG_BYTEIMA 103`
  - `#define TCPMSG_FLOATIMA 104`
  - `#define TCPMSG_FLOATIMASET 105`
  - `#define TCPMSG_INT32 106`
  - `#define TCPMSG_FLOAT 107`
  - `#define TCPMSG_DOUBLE 108`

### 11.432.1 Detailed Description

Direct message passing over TCP connections

Definition in file [TCPmessage.H](#).

### 11.432.2 Define Documentation

#### 11.432.2.1 `#define TCPMSG_END 0`

types of possible message fields:

Definition at line 53 of file TCPmessage.H.

## 11.433 test-align.C File Reference

```
#include "calibrateFunctions.H"  
#include "Image.H"  
#include "Image_ColorOps.H"  
#include "log.H"  
#include "Raster.H"  
#include <iostream>  
#include <stdio.h>
```

Include dependency graph for test-align.C:



### Functions

- int **main** (int argc, char \*\*argv)

#### 11.433.1 Detailed Description

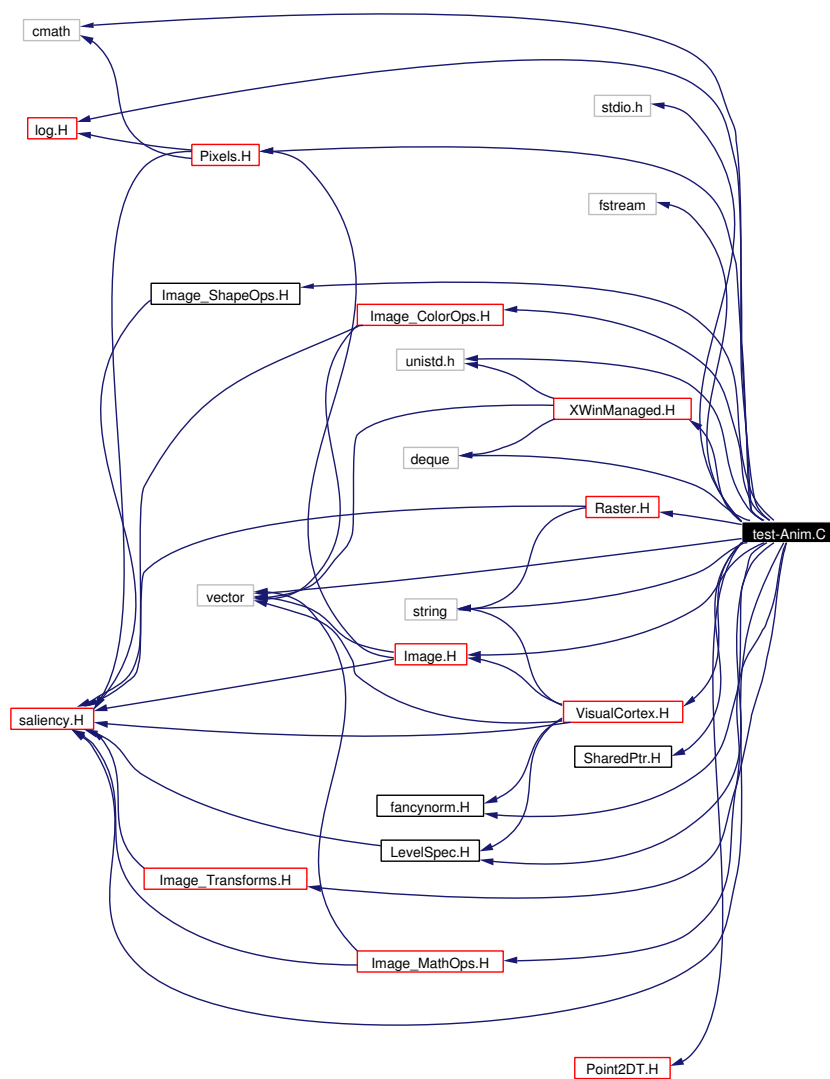
FIXME

Definition in file [test-align.C](#).

## 11.434 test-Anim.C File Reference

```
#include <cmath>
#include <unistd.h>
#include <stdio.h>
#include <vector>
#include <fstream>
#include <string>
#include <deque>
#include "saliency.H"
#include "log.H"
#include "Image.H"
#include "VisualCortex.H"
#include "Raster.H"
#include "Pixels.H"
#include "SharedPtr.H"
#include "LevelSpec.H"
#include "fancynorm.H"
#include "XWinManaged.H"
#include "Image_ColorOps.H"
#include "Image_Transforms.H"
#include "Image_ShapeOps.H"
#include "Image_MathOps.H"
#include "Point2DT.H"
```

Include dependency graph for test-Anim.C:



## Classes

- class **InputLine**
- struct **SomeInfo**
- class **Animate**

## Functions

- int **getFrameNumber** (double s)
- int **main** (void)

### 11.434.1 Detailed Description

FIXME

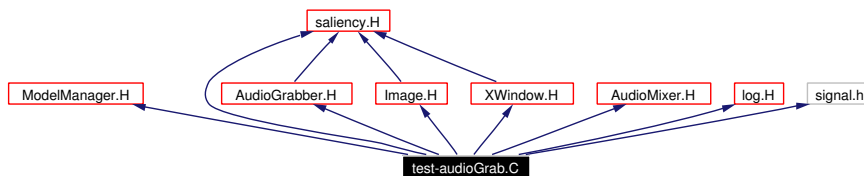
Definition in file [test-Anim.C](#).



## 11.435 test-audioGrab.C File Reference

```
#include "ModelManager.H"
#include "AudioGrabber.H"
#include "AudioMixer.H"
#include "Image.H"
#include "log.H"
#include "saliency.H"
#include "XWindow.H"
#include <signal.h>
```

Include dependency graph for test-audioGrab.C:



### Functions

- void [terminate](#) (int s)  
*Signal handler (e.g., for control-C).*
- int **main** (int argc, const char \*\*argv)

### Variables

- bool [goforever](#) = true  
*Will turn false on interrupt signal.*

#### 11.435.1 Detailed Description

Test audio digitizer

Definition in file [test-audioGrab.C](#).

## 11.435.2 Function Documentation

### 11.435.2.1 void terminate (int s)

Signal handler (e.g., for control-C).

Definition at line 49 of file test-audioGrab.C.

References goforever.

## 11.435.3 Variable Documentation

### 11.435.3.1 bool **goforever** = true [static]

Will turn false on interrupt signal.

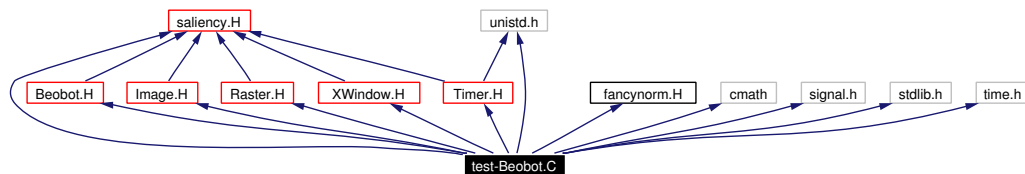
Definition at line 46 of file test-audioGrab.C.

Referenced by terminate().

## 11.436 test-Beobot.C File Reference

```
#include "Beobot.H"
#include "fancynorm.H"
#include "Image.H"
#include "Raster.H"
#include "Timer.H"
#include "saliency.H"
#include "XWindow.H"
#include <cmath>
#include <signal.h>
#include <stdlib.h>
#include <time.h>
#include <unistd.h>
```

Include dependency graph for test-Beobot.C:



## Functions

- int **main** (int argc, char \*\*argv)

### 11.436.1 Detailed Description

Test [Beobot](#) software

Definition in file [test-Beobot.C](#).

## 11.437 test-BeoChip.C File Reference

```
#include "BeoChip.H"
#include "ModelManager.H"
#include "ModelOptionDefs.H"
#include <cstdlib>
#include <iostream>
```

Include dependency graph for test-BeoChip.C:

### Classes

- class [MyBeoChipListener](#)  
*Our own little [BeoChipListener](#).*

### Functions

- int [main](#) (const int argc, const char \*argv[ ])  
*This program provides basic interfacing to the [BeoChip](#).*

#### 11.437.1 Detailed Description

test suite for Brian Hudson's [BeoChip](#)

Definition in file [test-BeoChip.C](#).

#### 11.437.2 Function Documentation

##### 11.437.2.1 int main (const int *argc*, const char \* *argv*[ ])

This program provides basic interfacing to the [BeoChip](#).

See the [BeoChip](#) class for details

Definition at line 82 of file test-BeoChip.C.

References [SharedPtr< T >::dynCastFrom\(\)](#), [main\(\)](#), [MYLOGVERB](#), and [randomUpTo\(\)](#).

## 11.438 test-beoGrab-master.C File Reference

```
#include "saliency.H"
#include "Image.H"
#include "Beowulf.H"
#include "XWindow.H"
#include "log.H"
#include "ModelManager.H"
#include <stdio>
#include <stdlib>
#include <string>
```

Include dependency graph for test-beoGrab-master.C:

### Defines

- `#define NAVG 20`

### Functions

- `int main (const int argc, const char **argv)`

#### 11.438.1 Detailed Description

Test frame grabbing and X display

Definition in file [test-beoGrab-master.C](#).

#### 11.438.2 Function Documentation

##### 11.438.2.1 `int main (const int argc, const char ** argv)`

This simple executable tests video frame grabbing through the video4linux driver (see [V4Lgrabber.H](#)) or the IEEE1394 (firewire) grabber (see [IEEE1394grabber.H](#)). Selection of the grabber type is made via the `-fg-type=XX` command-line option.

Definition at line 55 of file test-beoGrab-master.C.

References `TCPmessage::addInt32()`, `XWindow::drawImage()`, `TCPmessage::getElement()`, `main()`, and `TCPmessage::reset()`.

## 11.439 test-BeoSub.C File Reference

```
#include "BeoSub.H"  
#include "ModelManager.H"  
#include "BeoSub-defs.H"  
#include "Raster.H"
```

Include dependency graph for test-BeoSub.C:

### Enumerations

- enum **MajorMode** {  
    MM\_INIT, MM\_TEST, MM\_PANIC, MM\_GRAB,  
    MM\_GEOMETRY, MM\_WAITGO, MM\_PASSGATE, MM\_-  
    SEEKLIGHT,  
    MM\_SEEKBINS, MM\_SEEKHYDRO, MM\_END }

### Functions

- int [main](#) (const int argc, const char \*\*argv)  
*Simple test program for the [BeoSub](#) submarine.*

### 11.439.1 Detailed Description

Test [BeoSub](#) submarine basic functions

Definition in file [test-BeoSub.C](#).

### 11.439.2 Function Documentation

#### 11.439.2.1 int main (const int *argc*, const char \*\* *argv*)

Simple test program for the [BeoSub](#) submarine.

Simple test program for the [BeoSub](#) submarine

Definition at line 49 of file test-BeoSub.C.

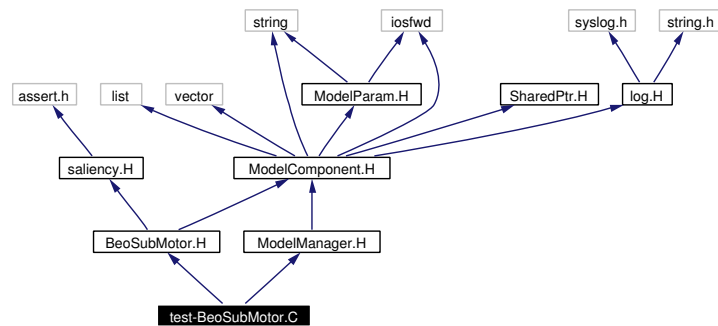
References [BEOSUB\\_CAM\\_FRONT](#), [centroid\(\)](#), [GATEDEPTH](#), [GATEPASSDEPTH](#), [Image< T >::getArrayPtr\(\)](#), [Image< T >::getHeight\(\)](#), [Image< T >::getWidth\(\)](#), [Point2D::i](#), [Point2D::j](#), [luminance\(\)](#), [main\(\)](#), and [sum\(\)](#).

## 11.440 test-BeoSubMotor.C File Reference

```
#include "BeoSubMotor.H"
```

```
#include "ModelManager.H"
```

Include dependency graph for test-BeoSubMotor.C:



### Functions

- int [main](#) (const int argc, const char \*\*argv)  
Simple test program for the [BeoSub](#) submarine.

#### 11.440.1 Detailed Description

Test [BeoSub](#) submarine motors

Definition in file [test-BeoSubMotor.C](#).

#### 11.440.2 Function Documentation

##### 11.440.2.1 int main (const int argc, const char \*\* argv)

Simple test program for the [BeoSub](#) submarine.

Simple test program for the [BeoSub](#) submarine

Definition at line 42 of file test-BeoSubMotor.C.

References [main\(\)](#).

## 11.441 test-binMorph.C File Reference

```
#include "Image.H"  
#include "log.H"  
#include "Raster.H"  
#include "XWinManaged.H"  
#include "Image_FilterOps.H"  
#include "Image_ColorOps.H"  
#include <iostream>
```

Include dependency graph for test-binMorph.C:

### Functions

- int **main** (int argc, char \*\*argv)

### 11.441.1 Detailed Description

FIXME

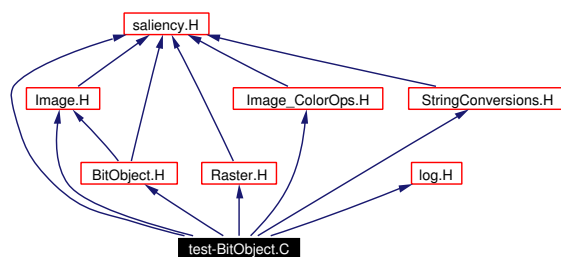
Definition in file [test-binMorph.C](#).



## 11.442 test-BitObject.C File Reference

```
#include "saliency.H"
#include "log.H"
#include "Image.H"
#include "BitObject.H"
#include "Raster.H"
#include "Image_ColorOps.H"
#include "StringConversions.H"
```

Include dependency graph for test-BitObject.C:



### Functions

- int **main** (const int argc, const char \*\*argv)

#### 11.442.1 Detailed Description

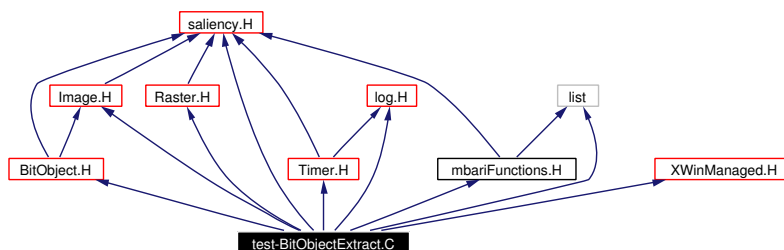
test program for the class [BitObject](#)

Definition in file [test-BitObject.C](#).

## 11.443 test-BitObjectExtract.C File Reference

```
#include "saliency.H"
#include "log.H"
#include "mbariFunctions.H"
#include "BitObject.H"
#include "Raster.H"
#include "Image.H"
#include "XWinManaged.H"
#include "Timer.H"
#include <list>
```

Include dependency graph for test-BitObjectExtract.C:



### Functions

- int **main** (const int argc, const char \*\*argv)

#### 11.443.1 Detailed Description

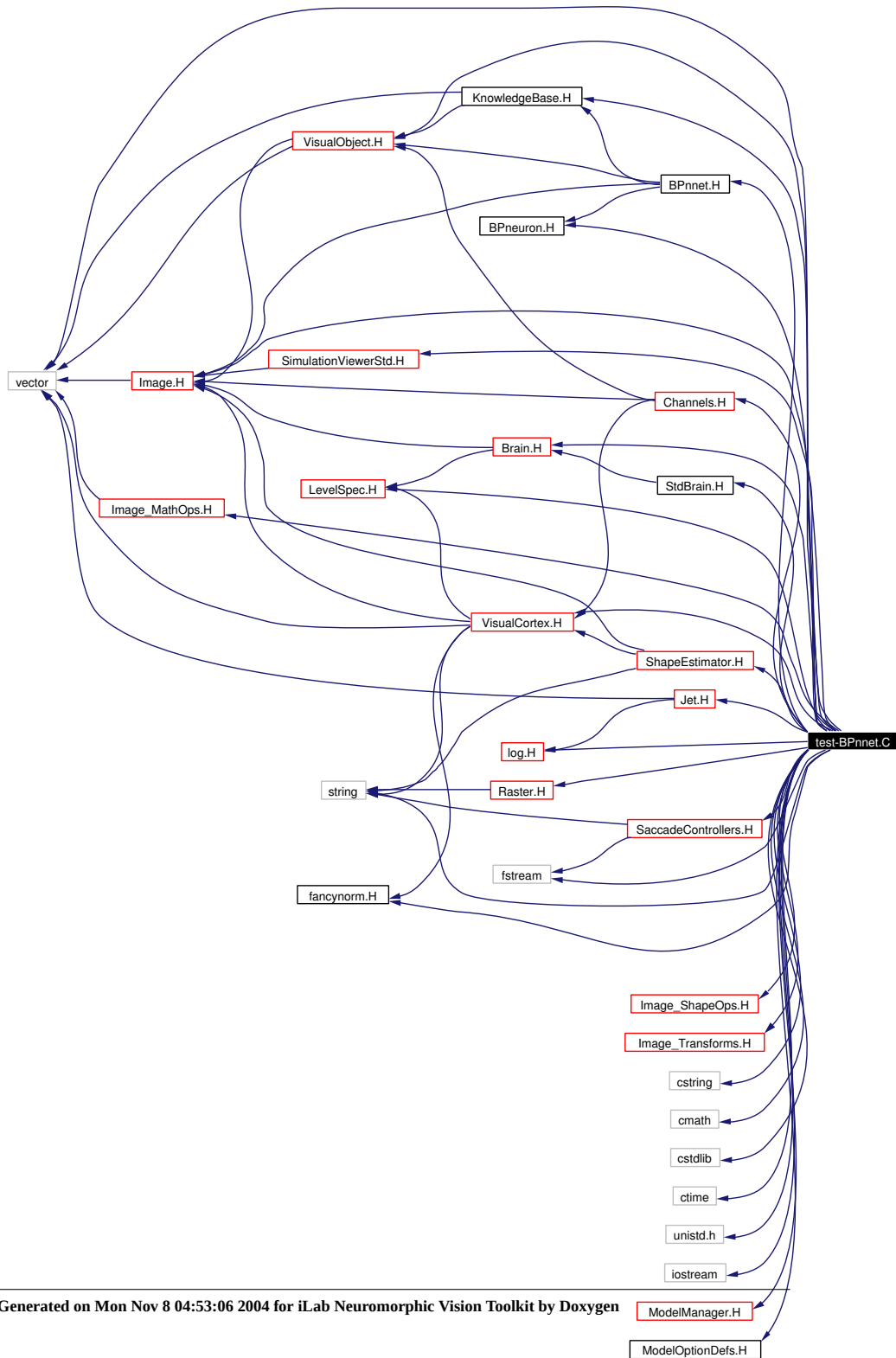
test the routine for extracting BitObjects

Definition in file [test-BitObjectExtract.C](#).

## 11.444 test-BPnnnet.C File Reference

```
#include "Brain.H"
#include "BPnnnet.H"
#include "BPneuron.H"
#include "KnowledgeBase.H"
#include "VisualObject.H"
#include "SaccadeControllers.H"
#include "Image.H"
#include "Image_ShapeOps.H"
#include "Image_Transforms.H"
#include "Image_MathOps.H"
#include "Jet.H"
#include "log.H"
#include "Raster.H"
#include "LevelSpec.H"
#include "Channels.H"
#include "fancynorm.H"
#include <string>
#include <cstring>
#include <cmath>
#include <fstream>
#include <vector>
#include <cstdlib>
#include <ctime>
#include <unistd.h>
#include <iostream>
#include "ModelManager.H"
#include "ModelOptionDefs.H"
#include "StdBrain.H"
#include "VisualCortex.H"
```

```
#include "ShapeEstimator.H"  
#include "SimulationViewerStd.H"  
Include dependency graph for test-BPnnnet.C:
```



## Defines

- `#define NHIDDEN 64`  
*number of hidden units*
- `#define NNEG 0 /*5*/`  
*number of negative samples to generate*
- `#define SRANGE 3`  
*sampling range in pixels for positive examples around most salient location*
- `#define SSTEP 3`  
*sampling step for positive examples, in pixels*

## Functions

- `int main (const int argc, const char **argv)`

### 11.444.1 Detailed Description

Test [BPnnnet](#) class

Definition in file [test-BPnnnet.C](#).

### 11.444.2 Define Documentation

#### 11.444.2.1 `#define NHIDDEN 64`

number of hidden units

Definition at line 71 of file [test-BPnnnet.C](#).

#### 11.444.2.2 `#define NNEG 0 /*5*/`

number of negative samples to generate

Definition at line 74 of file [test-BPnnnet.C](#).

#### 11.444.2.3 `#define SRANGE 3`

sampling range in pixels for positive examples around most salient location

Definition at line 78 of file [test-BPnnnet.C](#).

**11.444.2.4 #define SSTEP 3**

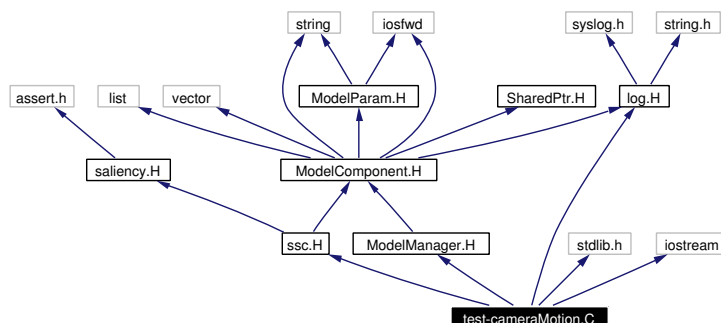
sampling step for positive examples, in pixels

Definition at line 81 of file test-BPnnnet.C.

## 11.445 test-cameraMotion.C File Reference

```
#include "ssc.H"
#include "log.H"
#include <stdlib.h>
#include <iostream>
#include "ModelManager.H"
```

Include dependency graph for test-cameraMotion.C:



### Defines

- #define **PAN** 2
- #define **TILT** 3
- #define **SERIAL** "/dev/ttyS0"

### Functions

- int **main** (const int argc, const char \*\*argv)

### Variables

- bool **panSuc**
- bool **TiltSuc**

### 11.445.1 Detailed Description

FIXME

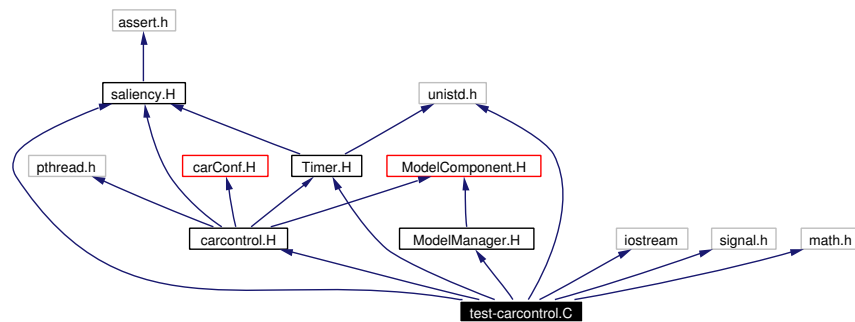


Definition in file [test-cameraMotion.C](#).

## 11.446 test-carcontrol.C File Reference

```
#include "saliency.H"
#include "carcontrol.H"
#include "ModelManager.H"
#include "Timer.H"
#include <iostream>
#include <unistd.h>
#include <signal.h>
#include <math.h>
```

Include dependency graph for test-carcontrol.C:



## Functions

- void [terminate](#) (int s)  
*Signal handler (e.g., for control-C).*
- int [main](#) (const int argc, const char \*\*argv)  
*The main method. Try running this on the [Beobot](#) to see how it behaves.*

## Variables

- bool [goforever](#) = true  
*Will turn false on interrupt signal.*

## 11.446.1 Detailed Description

Test the [CarControl](#) class

Definition in file [test-carcontrol.C](#).

## 11.446.2 Function Documentation

### 11.446.2.1 `int main (const int argc, const char ** argv)`

The main method. Try running this on the [Beobot](#) to see how it behaves.

Definition at line 56 of file test-carcontrol.C.

References [Timer::get\(\)](#), [goforever](#), [main\(\)](#), [Timer::reset\(\)](#), [SPEED\\_RAMP\\_LINEAR](#), [SPEED\\_RAMP\\_SIGMOID](#), and [terminate\(\)](#).

### 11.446.2.2 `void terminate (int s)`

Signal handler (e.g., for control-C).

Definition at line 49 of file test-carcontrol.C.

References [goforever](#).

## 11.446.3 Variable Documentation

### 11.446.3.1 `bool goforever = true` [`static`]

Will turn false on interrupt signal.

Definition at line 46 of file test-carcontrol.C.

Referenced by [main\(\)](#), and [terminate\(\)](#).

## 11.447 test-color.C File Reference

```
#include "saliency.H"
#include "ModelManager.H"
#include "ModelOptionDefs.H"
#include "Brain.H"
#include "Channels.H"
#include "VisualCortex.H"
#include "SaccadeControllers.H"
#include "SimulationViewerStd.H"
#include "ShapeEstimator.H"
#include "SaliencyMap.H"
#include "TaskRelevanceMap.H"
#include "AttentionGuidanceMap.H"
#include "WinnerTakeAll.H"
#include "Raster.H"
#include "log.H"
#include "FrameSeries.H"
#include <iostream>
#include "SingleChannel.H"
Include dependency graph for test-color.C:
```

### Functions

- int **main** (const int argc, const char \*\*argv)

#### 11.447.1 Detailed Description

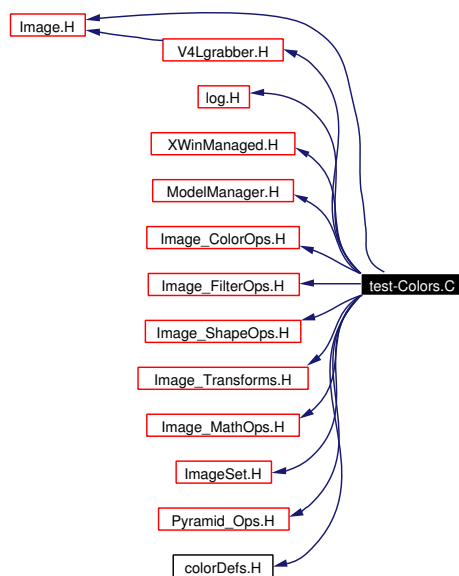
test whether single-opponent color channels work

Definition in file [test-color.C](#).

## 11.448 test-Colors.C File Reference

```
#include "Image.H"  
#include "log.H"  
#include "V4Lgrabber.H"  
#include "XWinManaged.H"  
#include "ModelManager.H"  
#include "Image_ColorOps.H"  
#include "Image_FilterOps.H"  
#include "Image_ShapeOps.H"  
#include "Image_Transforms.H"  
#include "Image_MathOps.H"  
#include "ImageSet.H"  
#include "Pyramid_Ops.H"  
#include "colorDefs.H"
```

Include dependency graph for test-Colors.C:



## Functions

- `int main (const int argc, const char **argv)`

### 11.448.1 Detailed Description

Import images from a video device and display the old and new ways to compute RG and BY color opponencies

Definition in file [test-Colors.C](#).

### 11.448.2 Function Documentation

#### 11.448.2.1 `int main (const int argc, const char ** argv)`

This program grabs a frame from a framegrabber and computes and displays a number of low-level features. This was done as a demo to explain the dissection of an image into features for a public outreach project.

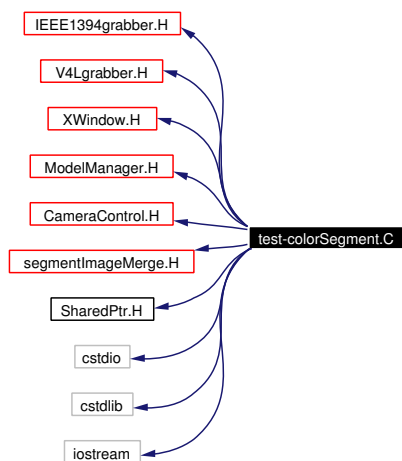
Definition at line 62 of file `test-Colors.C`.

References `CloseButtonListener::add()`, `Image< T >::clamp()`, `colorStain()`, `XWinManaged::drawImage()`, `FULLTRACE`, `SharedPtr< T >::get()`, `Image< T >::getDims()`, `getRGBY()`, `getRGBYsimple()`, `main()`, `CloseButtonListener::pressedAnyCloseButton()`, `takeMax()`, and `ZEROS`.

## 11.449 test-colorSegment.C File Reference

```
#include "IEEE1394grabber.H"  
#include "V4Lgrabber.H"  
#include "XWindow.H"  
#include "ModelManager.H"  
#include "CameraControl.H"  
#include "segmentImageMerge.H"  
#include "SharedPtr.H"  
#include <cstdio>  
#include <cstdlib>  
#include <iostream>
```

Include dependency graph for test-colorSegment.C:



### Defines

- #define NAVG 20

### Functions

- int **main** (const int argc, const char \*\*argv)

### 11.449.1 Detailed Description

Test IEEE1394 frame grabbing and X display

Definition in file [test-colorSegment.C](#).



## 11.450 test-Compare.C File Reference

```
#include "Image.H"  
#include "log.H"  
#include "Raster.H"  
#include "XWindow.H"  
#include <cstdlib>
```

Include dependency graph for test-Compare.C:

### Functions

- int **main** (int argc, char \*\*argv)

### 11.450.1 Detailed Description

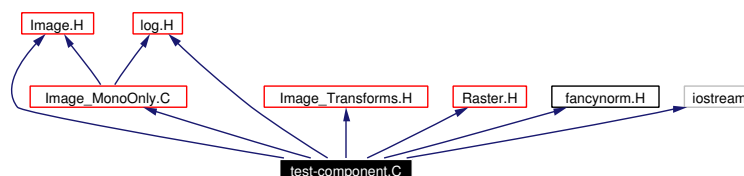
FIXME

Definition in file [test-Compare.C](#).

## 11.451 test-component.C File Reference

```
#include "Image.H"  
#include "Image_Transforms.H"  
#include "log.H"  
#include "Raster.H"  
#include "fancynorm.H"  
#include "Image_MonoOnly.C"  
#include <iostream>
```

Include dependency graph for test-component.C:



### Functions

- `int main (int argc, char **argv)`

#### 11.451.1 Detailed Description

FIXME

Definition in file [test-component.C](#).

## 11.452 test-compress.C File Reference

```
#include "saliency.H"
#include "log.H"
#include "Image.H"
#include "VisualCortex.H"
#include "Brain.H"
#include "Channels.H"
#include "Raster.H"
#include "Pixels.H"
#include "SharedPtr.H"
#include "LevelSpec.H"
#include "fancynorm.H"
#include "XWinManaged.H"
#include "Image_All.H"
#include "Image_ColorOps.H"
#include "Image_Transforms.H"
#include "Image_ShapeOps.H"
#include "Image_MathOps.H"
#include "SaccadeControllers.H"
#include "Image_DrawOps.H"
#include "Point2D.H"
#include "ModelManager.H"
#include "ShapeEstimator.H"
#include <sys/stat.h>
#include <sys/types.h>
#include <iostream>
#include <fstream>
#include <list>
#include <deque>
#include <stdio.h>
```

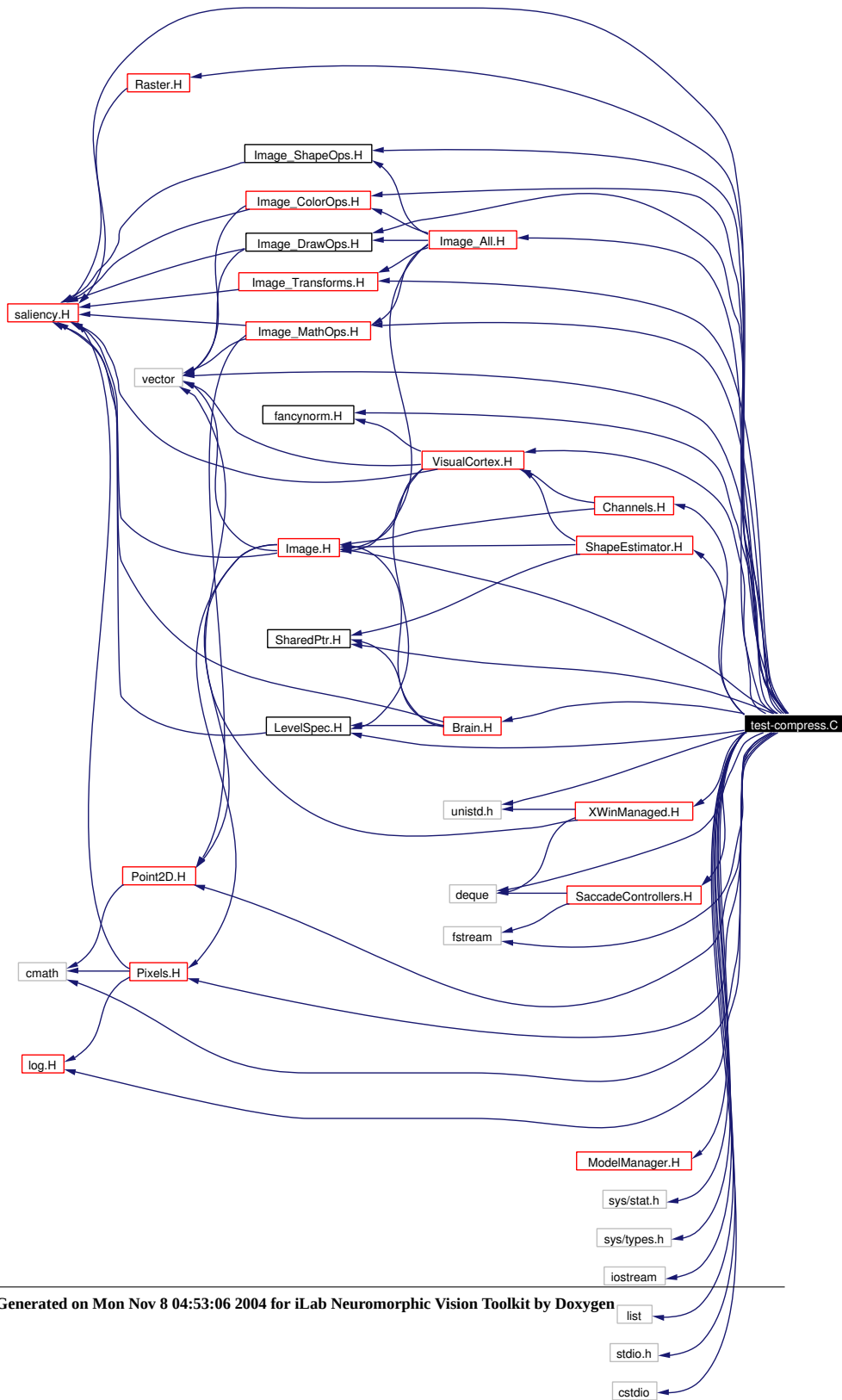
```
#include <unistd.h>
```

```
#include <cmath>
```

```
#include <stdio>
```

```
#include <vector>
```

Include dependency graph for test-compress.C:



## Defines

- `#define SQUARE(x) ((x)*(x))`

## Typedefs

- `typedef void(* PF )(int, char **)`

## Functions

- float **gaussian2d** (float x, float y, float xrad, float yrad)
- void **calc\_pixel\_value** (Image< PixRGB< byte > > &image, int x, int y, float theta, PixRGB< byte > &val)
- float **tsf** (float ecc)
- float **max\_size\_at\_pixel** (Point2D salient, int x, int y)
- float **max\_size\_at\_pixel** (float ecc)
- void **foveate** (Image< PixRGB< byte > > &in, Image< PixRGB< byte > > &out, Point2D &sal)
- void **foveate** (Image< PixRGB< byte > > &in, Image< PixRGB< byte > > &out, Image< int > &dm)
- int **distmap\_4ssed** (Image< int > &img)
- void **do\_shapeestimator\_compress** (int argc, char \*\*argv)
- void **WriteInt** (const Image< int > &image, const char \*fname,...)
- Image< int > **ReadInt** (const char \*fname,...)
- void **do\_salmap\_prediction\_compress** (int argc, char \*argv[ ])
- void **do\_salmap\_compress** (int argc, char \*argv[ ])
- void **do\_saliency\_compress** (int argc, char \*\*argv)
- int **main** (int argc, char \*\*argv)

## Variables

- double **dpp**
- float **sigma** = 4.0f

### 11.452.1 Detailed Description

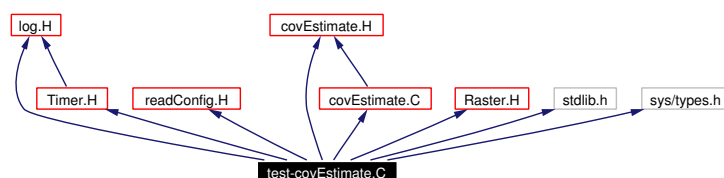
FIXME

Definition in file [test-compress.C](#).

## 11.453 test-covEstimate.C File Reference

```
#include "log.H"
#include "readConfig.H"
#include "covEstimate.H"
#include "covEstimate.C"
#include "Raster.H"
#include "Timer.H"
#include <stdlib.h>
#include <sys/types.h>
```

Include dependency graph for test-covEstimate.C:



### Defines

- #define **TYPE** float

### Functions

- int **main** (int argc, char \*argv[ ])

### Variables

- char \* **configFile**  
*This is the configFile name.*
- **PixRGB**< float > **pix**  
*generic pixel*
- int **itemNumber**  
*number of items if training*

### 11.453.1 Detailed Description

quick estimator for covariance matrix

Definition in file [test-covEstimate.C](#).

### 11.453.2 Variable Documentation

#### 11.453.2.1 `char* configFile`

This is the configFile name.

Definition at line 61 of file test-covEstimate.C.

#### 11.453.2.2 `int itemNumber`

number of items if training

Definition at line 67 of file test-covEstimate.C.

Referenced by `main()`, `readConfig::readFileTrue()`, `readConfig::readFileValueC()`, `readConfig::readFileValueF()`, `readConfig::readFileValueI()`, `readConfig::readFileValueName()`, `readConfig::readFileValueNameC()`, `readConfig::readFileValueNameF()`, `readConfig::readFileValueS()`, and `readConfig::setFileValue()`.

#### 11.453.2.3 `PixRGB<float> pix`

generic pixel

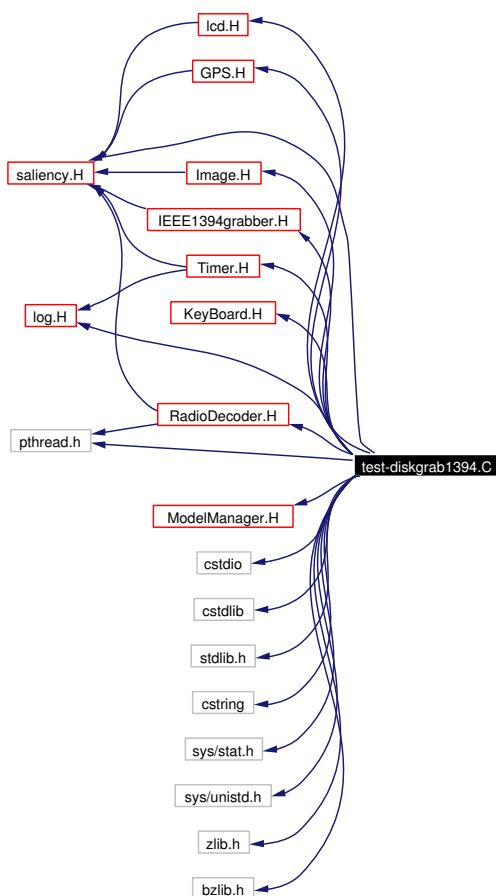
Definition at line 65 of file test-covEstimate.C.



## 11.454 test-diskgrab1394.C File Reference

```
#include "saliency.H"
#include "Image.H"
#include "IEEE1394grabber.H"
#include "Timer.H"
#include "lcd.H"
#include "KeyBoard.H"
#include "log.H"
#include "GPS.H"
#include "RadioDecoder.H"
#include "ModelManager.H"
#include <stdio>
#include <stdlib>
#include <stdlib.h>
#include <cstring>
#include <sys/stat.h>
#include <sys/unistd.h>
#include <zlib.h>
#include <bzlib.h>
#include <pthread.h>
```

Include dependency graph for test-diskgrab1394.C:



## Defines

- `#define NAVG 30`
- `#define OUTPATH "/home/tmp/7/diskgrab1394/"`
- `#define LINE_1 0`
- `#define LINE_2 64`
- `#define LINE_3 20`
- `#define LINE_4 84`

## Functions

- `SharedPtr< GpsDev > gps (NULL)`
- `SharedPtr< RadioDecoder > rd (NULL)`
- `void calibrate (ModelManager &mgr, SharedPtr< lcd > lcdDisp)`

- void **gzWriteFrame** (int frameNum, [Image](#)< [PixRGB](#)< [byte](#) > > &imag)
- void **bzWriteFrame** (int frameNum, [Image](#)< [PixRGB](#)< [byte](#) > > &imag)
- void \* **gpsGetThread** (void \*junk)
- void \* **rcGetThread** (void \*junk)
- void \* **imageWriterThread** (void \*junk)
- void **grabCycle** ([KeyBoard](#) &myKB, [SharedPtr](#)< [lcd](#) > lcdDisp, [SharedPtr](#)< [FrameGrabber](#) > &gb)
- int **main** (const int argc, const char \*\*argv)

## Variables

- char **dir** [10]
- bool **doWrite**
- bool **grabOk**
- bool **doQuit**
- bool **quitGrab**
- pthread\_mutex\_t **mutexGPS**
- pthread\_mutex\_t **mutexRC**
- [Image](#)< [PixRGB](#)< [byte](#) > > **ima**
- bool **gZip** = false bZip = false
- bool **wGPS** = false
- bool **wRC** = false
- float **accel**
- float **steer**
- int **frame**
- GPSPvtData **pvt**
- bool **beep** = true

### 11.454.1 Detailed Description

Test IEEE1394 frame grabbing and X display

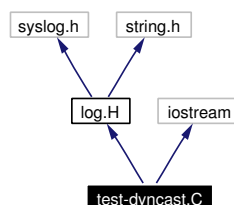
Definition in file [test-diskgrab1394.C](#).

## 11.455 test-dyncast.C File Reference

```
#include "log.H"
```

```
#include <iostream>
```

Include dependency graph for test-dyncast.C:



### Classes

- class **A**
- class **B**

### Functions

- int **main** (int argc, char \*\*argv)

### 11.455.1 Detailed Description

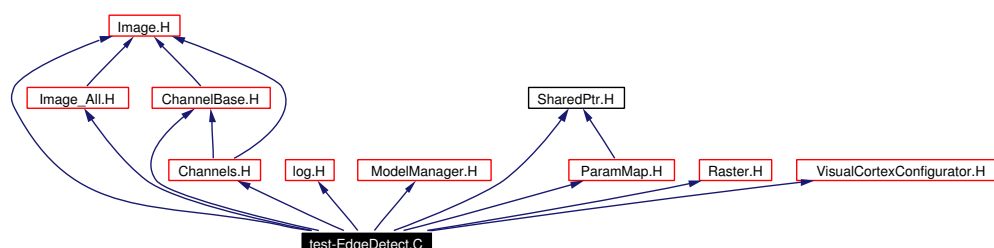
FIXME

Definition in file [test-dyncast.C](#).

## 11.456 test-EdgeDetect.C File Reference

```
#include "Image.H"
#include "Image_All.H"
#include "log.H"
#include "ModelManager.H"
#include "ParamMap.H"
#include "Raster.H"
#include "SharedPtr.H"
#include "ChannelBase.H"
#include "Channels.H"
#include "VisualCortexConfigurator.H"
```

Include dependency graph for test-EdgeDetect.C:



## Functions

- int **main** (const int argc, const char \*argv[ ])

### 11.456.1 Detailed Description

Definition in file [test-EdgeDetect.C](#).

## 11.457 test-FFN.C File Reference

```
#include "Raster.H"
#include "XWindow.H"
#include "XWinManaged.H"
#include "saliency.H"
#include "ModelManager.H"
#include "ComplexChannel.H"
#include "Channels.H"
#include "VisualCortex.H"
#include "FrameSeries.H"
#include "ModelOptionDefs.H"
#include "StdBrain.H"
#include "Jet.H"
#include "ShapeEstimator.H"
#include "Image.H"
#include "Pyramid_Ops.H"
#include "Image_MathOps.H"
#include "Image_ShapeOps.H"
#include "Image_FilterOps.H"
#include "Image_ColorOps.H"
#include "Image_Transforms.H"
#include "ImageSet.H"
#include "PyrBuilder.H"
#include "Point2D.H"
#include "Dims.H"
#include <cstdio>
#include <cstdlib>
#include <cstring>
#include <stdio.h>
#include <iostream>
```

```
#include <math.h>
#include <stdlib.h>
#include "FFN.H"
#include "ModelFace.H"
```

Include dependency graph for test-FFN.C:

## Defines

- #define [LEARN\\_RATE](#) 0.5
- #define [TRAIN\\_START\\_INDEX](#) 277
- #define [NUM\\_TRAIN\\_SAMPLE](#) 200
- #define [TEST\\_START\\_INDEX](#) 234
- #define [NUM\\_TEST\\_SAMPLE](#) 100

## Functions

- [Image](#)< float > [getPartImage](#) (char \*iName, [Rectangle](#) pr)
- [Image](#)< float > [getPartImage](#) ([Image](#)< float > img, [Rectangle](#) pr)
- [Rectangle](#) [getPartRect](#) (char \*iName, char \*dName, FacePart part, int opp)
- [Rectangle](#) [getPartRect](#) ([Image](#)< float > img, char \*dName, FacePart part, int opp)
- void [getFeature](#) ([Rectangle](#) r, FacePart part, double \*features)
- void [getEyeFeature](#) ([Rectangle](#) r, double \*features)
- void [getNoseFeature](#) ([Rectangle](#) r, double \*features)
- void [getMouthFeature](#) ([Rectangle](#) r, double \*features)
- void [train](#) ([FeedForwardNetwork](#) \*ffn, FacePart part, [SharedPtr](#)< [StdBrain](#) > brain)
- void [fillEyeData](#) (int start, double \*\*pData, double \*\*nData, int \*pCt, int \*nCt)
- void [fillNoseData](#) (int start, double \*\*pData, double \*\*nData, int \*pCt, int \*nCt)
- void [fillMouthData](#) (int start, double \*\*pData, double \*\*nData, int \*pCt, int \*nCt)
- int [max](#) (int x, int y)
- int [min](#) (int x, int y)
- bool [within](#) ([Rectangle](#) r, int x, int y)
- void [printRegion](#) ([Image](#)< float > img, int sX, int eX, int dX, int sY, int eY, int dY)
- int [main](#) (const int argc, const char \*\*argv)

## Variables

- [CloseButtonListener](#) **wList**
- [XWinManaged](#) \* **imgWin**
- [Image](#)< [byte](#) > **targetmask**
- [SharedPtr](#)< [VisualCortex](#) > **vc**
- [SharedPtr](#)< [ShapeEstimator](#) > **se**
- [SharedPtr](#)< [OrientationChannel](#) > **oriChan**
- [Image](#)< [float](#) > **currImg**
- [ImageSet](#)< [float](#) > **currGaPyr** [NUM\_DIR]
- [XWinManaged](#) \* **iWin**
- [XWinManaged](#) \* **gaWin** [NUM\_DIR][3]
- [int](#) **start\_win** = 1

### 11.457.1 Detailed Description

testing Feed Forward Network class for use to train any mapping

Definition in file [test-FFN.C](#).

### 11.457.2 Define Documentation

#### 11.457.2.1 #define LEARN\_RATE 0.5

This class test implementation a simple feedforward network with backpropagation.

Definition at line 80 of file [test-FFN.C](#).



## 11.458 test-flood.C File Reference

```
#include "Image.H"
#include "Image_ColorOps.H"
#include "Image_Transforms.H"
#include "Timer.H"
#include "log.H"
#include "Raster.H"
#include <iostream>
#include <cstdio>
#include <vector>
```

Include dependency graph for test-flood.C:

### Functions

- int **main** (int argc, char \*\*argv)

### 11.458.1 Detailed Description

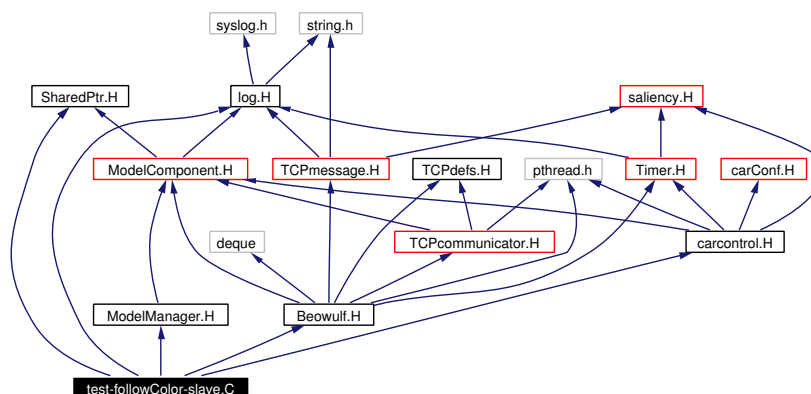
FIXME

Definition in file [test-flood.C](#).

## 11.459 test-followColor-slave.C File Reference

```
#include "log.H"
#include "ModelManager.H"
#include "Beowulf.H"
#include "carcontrol.H"
#include "SharedPtr.H"
```

Include dependency graph for test-followColor-slave.C:



## Functions

- int [main](#) (const int argc, const char \*\*argv)

### 11.459.1 Detailed Description

Test color segment following - slave node

Definition in file [test-followColor-slave.C](#).

### 11.459.2 Function Documentation

#### 11.459.2.1 int main (const int argc, const char \*\* argv)

The main routine. Receive signals from master node and performs requested actions.

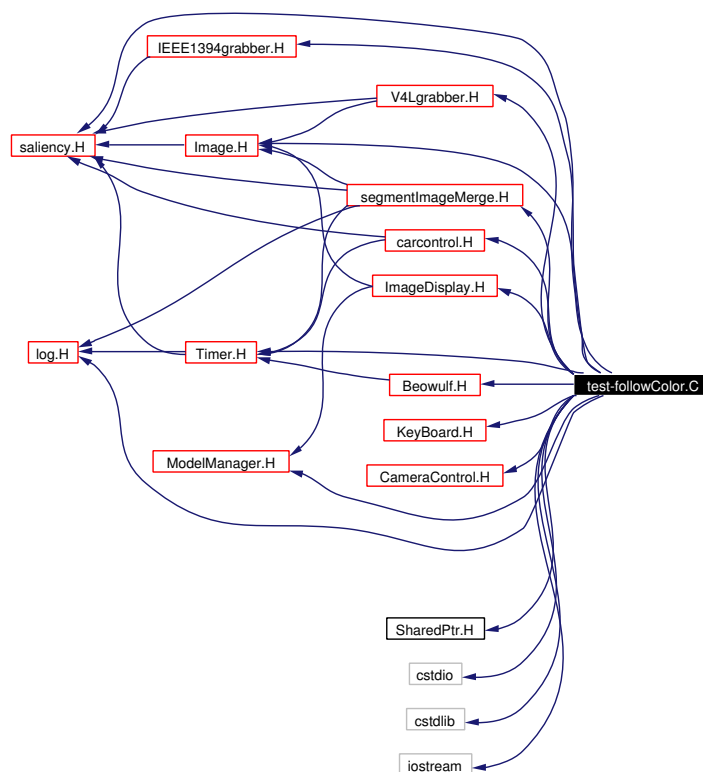
Definition at line 45 of file test-followColor-slave.C.

References TCPmessage::getElement(), and main().

## 11.460 test-followColor.C File Reference

```
#include "saliency.H"  
#include "Image.H"  
#include "IEEE1394grabber.H"  
#include "V4Lgrabber.H"  
#include "Timer.H"  
#include "log.H"  
#include "segmentImageMerge.H"  
#include "ModelManager.H"  
#include "Beowulf.H"  
#include "KeyBoard.H"  
#include "CameraControl.H"  
#include "carcontrol.H"  
#include "ImageDisplay.H"  
#include "SharedPtr.H"  
#include <cstdio>  
#include <cstdlib>  
#include <iostream>
```

Include dependency graph for test-followColor.C:



## Defines

- `#define NAVG 20`

## Functions

- `int main (const int argc, const char **argv)`  
*The main routine. Grab frames, process, send commands to slave node.*

### 11.460.1 Detailed Description

Test color segment following - master node

Definition in file [test-followColor.C](#).

## 11.460.2 Function Documentation

### 11.460.2.1 `int main (const int argc, const char ** argv)`

The main routine. Grab frames, process, send commands to slave node.

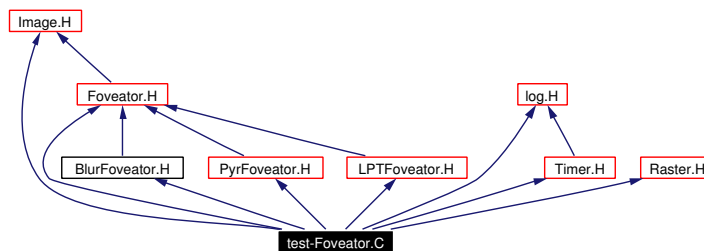
Definition at line 59 of file test-followColor.C.

References `decXY()`, `Image< PixRGB< byte > >::drawCircle()`, `Image< PixRGB< byte > >::drawRect()`, `Timer::get()`, `SharedPtr< T >::get()`, `KeyBoard::getKey()`, `main()`, `NAVG`, `Timer::reset()`, `sqrt()`, and `uint64`.

## 11.461 test-Foveator.C File Reference

```
#include "Image.H"
#include "log.H"
#include "Raster.H"
#include "Timer.H"
#include "Foveator.H"
#include "BlurFoveator.H"
#include "PyrFoveator.H"
#include "LPTFoveator.H"
```

Include dependency graph for test-Foveator.C:



## Functions

- int [main](#) (int argc, char \*\*argv)  
*The main routine. Read images, process, output results.*

### 11.461.1 Detailed Description

Test the [Foveator](#) class

Definition in file [test-Foveator.C](#).

### 11.461.2 Function Documentation

#### 11.461.2.1 int main (int argc, char \*\* argv)

The main routine. Read images, process, output results.

Definition at line 51 of file test-Foveator.C.

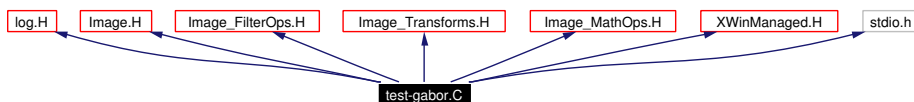
References LPTFoveator::foveate(), PyrFoveator::foveate(), BlurFoveator::foveate(), Timer::get(), Image< T >::getHeight(), LPTFoveator::getLPT(), Image< T >::getWidth(), Image< T >::initialized(), main(), Raster::ReadRGB(), Timer::reset(), PyrFoveator::setBaseRect(), and Raster::WriteRGB().



## 11.462 test-gabor.C File Reference

```
#include "log.H"  
#include "Image.H"  
#include "Image_FilterOps.H"  
#include "Image_Transforms.H"  
#include "Image_MathOps.H"  
#include "XWinManaged.H"  
#include <stdio.h>
```

Include dependency graph for test-gabor.C:



### Functions

- int **main** (int argc, char \*\*argv)

#### 11.462.1 Detailed Description

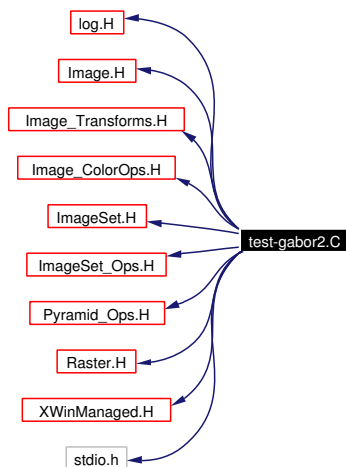
FIXME

Definition in file [test-gabor.C](#).

## 11.463 test-gabor2.C File Reference

```
#include "log.H"  
#include "Image.H"  
#include "Image_Transforms.H"  
#include "Image_ColorOps.H"  
#include "ImageSet.H"  
#include "ImageSet_Ops.H"  
#include "Pyramid_Ops.H"  
#include "Raster.H"  
#include "XWinManaged.H"  
#include <stdio.h>
```

Include dependency graph for test-gabor2.C:



### Defines

- #define **OPTION**(x) flags[c] = x; names[c] = #x; ++c;

### Functions

- int **main** (int argc, char \*\*argv)

### 11.463.1 Detailed Description

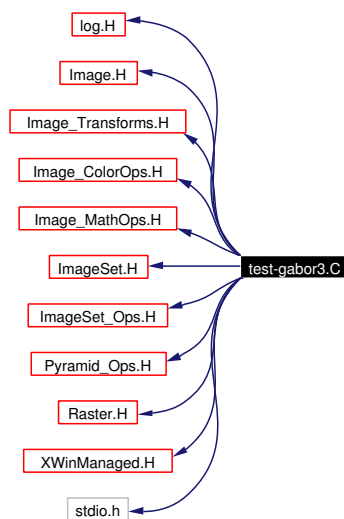
FIXME

Definition in file [test-gabor2.C](#).

## 11.464 test-gabor3.C File Reference

```
#include "log.H"  
#include "Image.H"  
#include "Image_Transforms.H"  
#include "Image_ColorOps.H"  
#include "Image_MathOps.H"  
#include "ImageSet.H"  
#include "ImageSet_Ops.H"  
#include "Pyramid_Ops.H"  
#include "Raster.H"  
#include "XWinManaged.H"  
#include <stdio.h>
```

Include dependency graph for test-gabor3.C:



## Functions

- int **main** (int argc, char \*\*argv)

### 11.464.1 Detailed Description

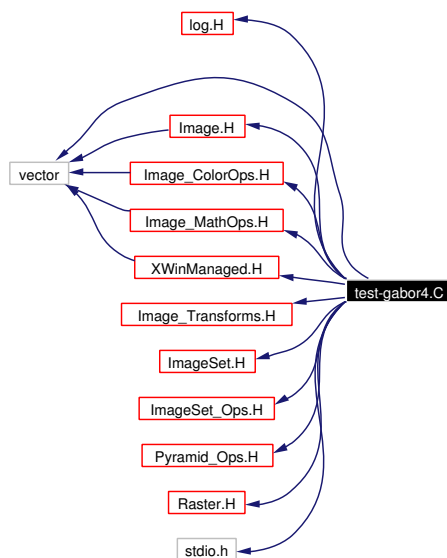
FIXME

Definition in file [test-gabor3.C](#).

## 11.465 test-gabor4.C File Reference

```
#include "log.H"
#include "Image.H"
#include "Image_Transforms.H"
#include "Image_ColorOps.H"
#include "Image_MathOps.H"
#include "ImageSet.H"
#include "ImageSet_Ops.H"
#include "Pyramid_Ops.H"
#include "Raster.H"
#include "XWinManaged.H"
#include <stdio.h>
#include <vector>
```

Include dependency graph for test-gabor4.C:



## Functions

- int **main** (int argc, char \*\*argv)

### 11.465.1 Detailed Description

FIXME

Definition in file [test-gabor4.C](#).

## 11.466 test-gist.C File Reference

```
#include "Raster.H"
#include "MPEGStream.H"
#include "XWindow.H"
#include "XWinManaged.H"
#include "saliency.H"
#include "ModelManager.H"
#include "ComplexChannel.H"
#include "Channels.H"
#include "VisualCortex.H"
#include "FrameSeries.H"
#include "ModelOptionDefs.H"
#include "StdBrain.H"
#include "Jet.H"
#include "ShapeEstimator.H"
#include "GistEstimator.H"
#include "Image.H"
#include "Pyramid_Ops.H"
#include "Image_MathOps.H"
#include "Image_ShapeOps.H"
#include "Image_FilterOps.H"
#include "Image_ColorOps.H"
#include "Image_Transforms.H"
#include "ImageSet.H"
#include "PyrBuilder.H"
#include "Point2D.H"
#include "Dims.H"
#include <stdio>
#include <stdlib>
#include <string>
```



```
#include <stdio.h>
#include <iostream>
#include <math.h>
#include "gistParams.H"
```

Include dependency graph for test-gist.C:

## Functions

- double \* **getFeatureVector** ([SharedPtr](#)< [StdBrain](#) > brain)
- void **getSubSum** (double \*res, [Image](#)< float > img)
- void **drawHistogram** (double \*arr)
- void **saveData** (double \*data, const char \*fName)
- [Image](#)< float > **normalize** ([Image](#)< float > img)
- void **printRegion** ([Image](#)< float > img, int sX, int eX, int dX, int sY, int eY, int dY)
- int **main** (const int argc, const char \*\*argv)

## Variables

- [CloseButtonListener wList](#)

### 11.466.1 Detailed Description

extract gist using available features of the image

Definition in file [test-gist.C](#).

### 11.466.2 Variable Documentation

#### 11.466.2.1 [CloseButtonListener wList](#)

display gist information

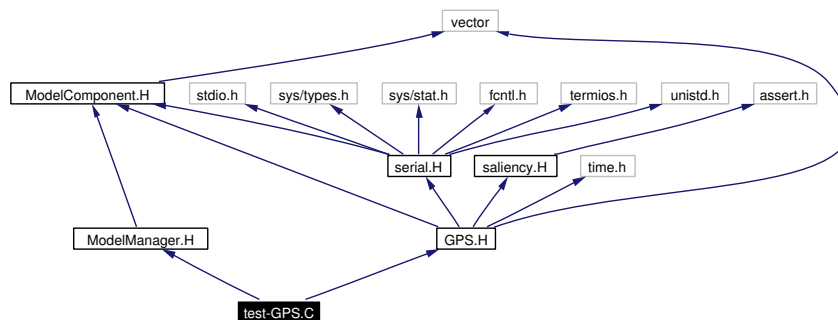
Definition at line 79 of file test-gist.C.

## 11.467 test-GPS.C File Reference

```
#include "GPS.H"
```

```
#include "ModelManager.H"
```

Include dependency graph for test-GPS.C:



## Functions

- int [main](#) (const int argc, const char \*\*argv)

*This program tests the GPS driver in class [GpsDev](#).*

### 11.467.1 Detailed Description

test GPS driver

Definition in file [test-GPS.C](#).

### 11.467.2 Function Documentation

#### 11.467.2.1 int main (const int argc, const char \*\* argv)

This program tests the GPS driver in class [GpsDev](#).

But I am afraid that only Nitin knows the details!

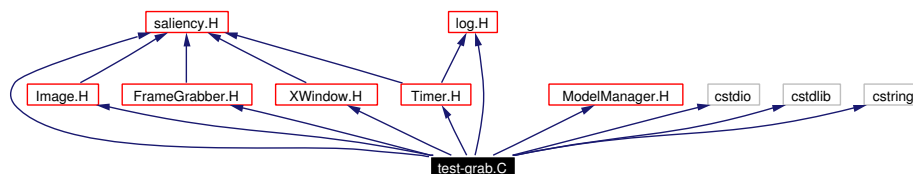
Definition at line 42 of file test-GPS.C.

References [main\(\)](#).

## 11.468 test-grab.C File Reference

```
#include "saliency.H"
#include "Image.H"
#include "FrameGrabber.H"
#include "Timer.H"
#include "XWindow.H"
#include "log.H"
#include "ModelManager.H"
#include <stdio>
#include <stdlib>
#include <string>
```

Include dependency graph for test-grab.C:



### Defines

- #define NAVG 20

### Functions

- int [main](#) (const int argc, const char \*\*argv)

#### 11.468.1 Detailed Description

Test frame grabbing and X display

Definition in file [test-grab.C](#).

## 11.468.2 Function Documentation

### 11.468.2.1 `int main (const int argc, const char ** argv)`

This simple executable tests video frame grabbing through the video4linux driver (see [V4Lgrabber.H](#)) or the IEEE1394 (firewire) grabber (see [IEEE1394grabber.H](#)). Selection of the grabber type is made via the `-fg-type=XX` command-line option.

Definition at line 55 of file `test-grab.C`.

References `XWindow::drawImage()`, `Timer::get()`, `SharedPtr< T >::get()`, `main()`, `NAVG`, `Timer::reset()`, and `uint64`.

## 11.469 test-gyro.C File Reference

```
#include "Gyro.H"
```

```
#include <iostream>
```

```
#include <unistd.h>
```

Include dependency graph for test-gyro.C:

### Functions

- int **main** ()

#### 11.469.1 Detailed Description

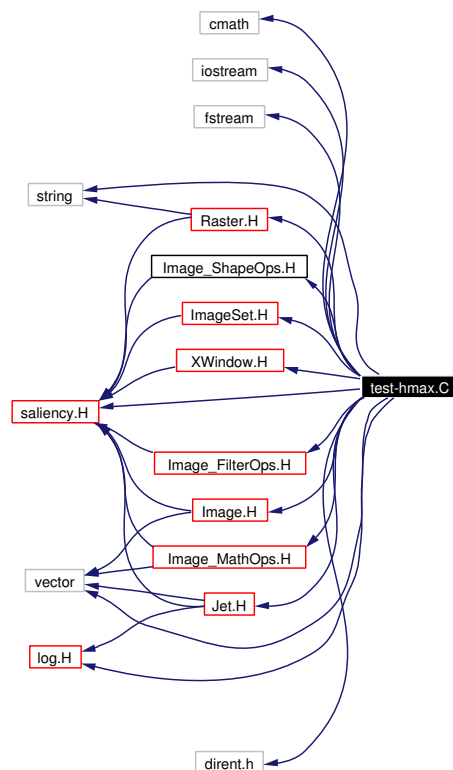
FIXME

Definition in file [test-gyro.C](#).

## 11.470 test-hmax.C File Reference

```
#include <cmath>
#include <iostream>
#include <fstream>
#include <string>
#include <vector>
#include <dirent.h>
#include "saliency.H"
#include "log.H"
#include "Image.H"
#include "Image_FilterOps.H"
#include "Image_ShapeOps.H"
#include "Image_MathOps.H"
#include "ImageSet.H"
#include "Raster.H"
#include "Jet.H"
#include "XWindow.H"
```

Include dependency graph for test-hmax.C:



## Functions

- float **sqr** (float x)
- vector< [string](#) > **readDir** ([string](#) inName)
- int [main](#) (int argc, char \*\*argv)

*The main routine. Read images, process, display results.*

### 11.470.1 Detailed Description

This is a test of the C++ hmax object recognition

Definition in file [test-hmax.C](#).

## 11.470.2 Function Documentation

### 11.470.2.1 `int main (int argc, char ** argv)`

The main routine. Read images, process, display results.

Definition at line 86 of file test-hmax.C.

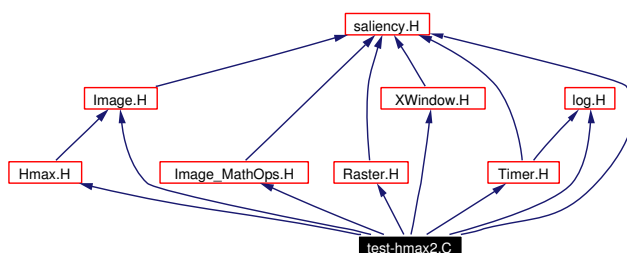
References `concatArray()`, `XWindow::drawImage()`, `fileList`, `Image< T >::getHeight()`, `Image< T >::getSize()`, `Image< T >::getVal()`, `main()`, `mean()`, `NO_INIT`, `Image< T >::normalize()`, `Raster::ReadGray()`, `rescale()`, `Image< T >::resize()`, `Image< T >::setVal()`, `sum()`, and `Raster::Visu()`.



## 11.471 test-hmax2.C File Reference

```
#include "saliency.H"
#include "Image.H"
#include "Image_MathOps.H"
#include "Hmax.H"
#include "Raster.H"
#include "log.H"
#include "Timer.H"
#include "XWindow.H"
```

Include dependency graph for test-hmax2.C:



### Defines

- `#define NORI 4`

### Functions

- `int main (int argc, char **argv)`

### Variables

- `int tItr = 1`
- `int c2Size = 0`
- `int target = 0`
- `string targetS1 = "tri"`
- `string targetS2 = "Tri"`
- `int imgHeight = 0`

- int **imgWidth** = 0
- int **trnFlag** = 0
- float **w** [ ] = {1,1}

### 11.471.1 Detailed Description

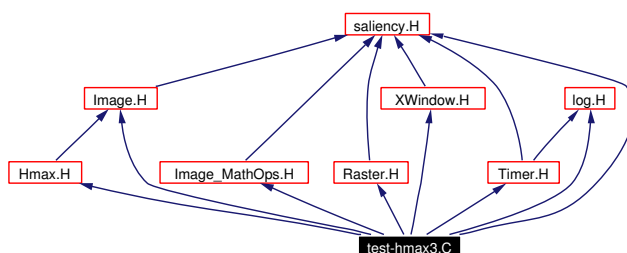
Test [Hmax](#) class and compare to original code

Definition in file [test-hmax2.C](#).

## 11.472 test-hmax3.C File Reference

```
#include "saliency.H"
#include "Image.H"
#include "Image_MathOps.H"
#include "Hmax.H"
#include "Raster.H"
#include "log.H"
#include "Timer.H"
#include "XWindow.H"
```

Include dependency graph for test-hmax3.C:



### Defines

- `#define NORI 4`

### Functions

- `int main (int argc, char **argv)`

### Variables

- `int tItr = 1`
- `int c2Size = 0`
- `int target = 0`
- `string targetS1 = "tri"`
- `string targetS2 = "Tri"`
- `int imgHeight = 0`

- int **imgWidth** = 0
- int **mixRate** = 5

### 11.472.1 Detailed Description

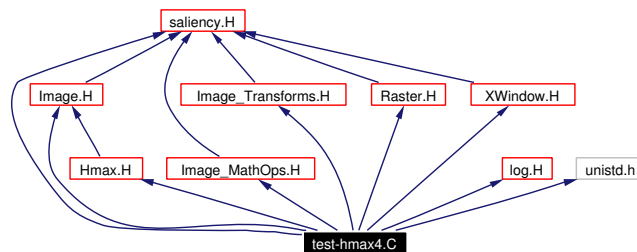
Test [Hmax](#) class and compare to original code

Definition in file [test-hmax3.C](#).

## 11.473 test-hmax4.C File Reference

```
#include "saliency.H"
#include "Image.H"
#include "Image_MathOps.H"
#include "Image_Transforms.H"
#include "Hmax.H"
#include "Raster.H"
#include "log.H"
#include "XWindow.H"
#include <unistd.h>
```

Include dependency graph for test-hmax4.C:



### Defines

- `#define NORI 4`

### Functions

- `int main (const int argc, const char **argv)`

#### 11.473.1 Detailed Description

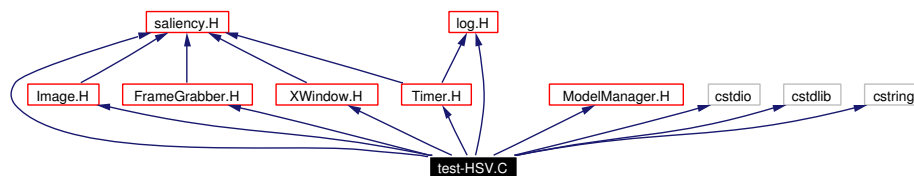
Test [Hmax](#) class and compare to original code

Definition in file [test-hmax4.C](#).

## 11.474 test-HSV.C File Reference

```
#include "saliency.H"
#include "Image.H"
#include "FrameGrabber.H"
#include "Timer.H"
#include "XWindow.H"
#include "log.H"
#include "ModelManager.H"
#include <stdio>
#include <stdlib>
#include <string>
```

Include dependency graph for test-HSV.C:



### Defines

- `#define NAVG 20`

### Functions

- `int main (const int argc, const char **argv)`

#### 11.474.1 Detailed Description

Test HSV histogramming of framegrabber input

Definition in file [test-HSV.C](#).

## 11.474.2 Function Documentation

### 11.474.2.1 `int main (const int argc, const char ** argv)`

This simple executable tests video frame grabbing through the video4linux driver (see [V4Lgrabber.H](#)) or the IEEE1394 (firewire) grabber (see [IEEE1394grabber.H](#)). Selection of the grabber type is made via the `-fg-type=XX` command-line option.

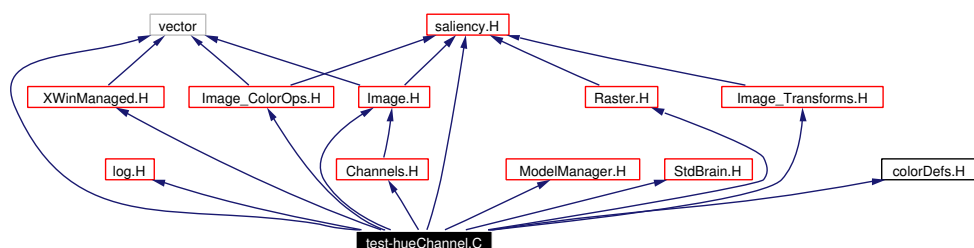
Definition at line 55 of file test-HSV.C.

References `XWindow::drawImage()`, `Image< PixRGB< byte > >::drawRect()`, `Image< T >::drawRect()`, `SharedPtr< T >::get()`, `Image< PixRGB< byte > >::getHeight()`, `PixRGB< float >::getHSV()`, `Image< PixRGB< byte > >::getVal()`, `Image< PixRGB< byte > >::getWidth()`, `main()`, `Image< T >::resize()`, `sizeX`, and `sizeY`.

## 11.475 test-hueChannel.C File Reference

```
#include "saliency.H"
#include "log.H"
#include "Image.H"
#include "XWinManaged.H"
#include "Channels.H"
#include "Raster.H"
#include "ModelManager.H"
#include "StdBrain.H"
#include "Image_ColorOps.H"
#include "Image_Transforms.H"
#include "colorDefs.H"
#include <vector>
```

Include dependency graph for test-hueChannel.C:



## Functions

- int **main** (const int argc, const char \*\*argv)

### 11.475.1 Detailed Description

Test the [HueChannel](#), in particular the [SkinHueChannel](#)

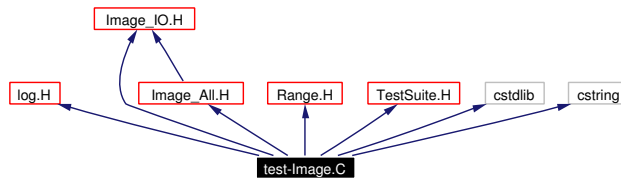
Definition in file [test-hueChannel.C](#).



## 11.476 test-Image.C File Reference

```
#include "log.H"
#include "Image_All.H"
#include "Image_IO.H"
#include "Range.H"
#include "TestSuite.H"
#include <cstdlib>
#include <cstring>
```

Include dependency graph for test-Image.C:



### Classes

- struct **InstanceCounter**

### Functions

- template<class T> **Image**< T > **sequenceImage** (int w, int h, int from=0)
- void **Image\_xx\_construct\_from\_array\_xx\_1** (**TestSuite** &suite)
- void **Image\_xx\_construct\_UDT\_xx\_1** (**TestSuite** &suite)
- void **Image\_xx\_construct\_and\_clear\_xx\_1** (**TestSuite** &suite)
- void **Image\_xx\_default\_construct\_xx\_1** (**TestSuite** &suite)
- void **Image\_xx\_swap\_xx\_1** (**TestSuite** &suite)
- void **Image\_xx\_copy\_xx\_1** (**TestSuite** &suite)
- void **Image\_xx\_copy\_on\_write\_xx\_1** (**TestSuite** &suite)
- void **Image\_xx\_copy\_on\_write\_xx\_2** (**TestSuite** &suite)
- void **Image\_xx\_copy\_on\_write\_xx\_3** (**TestSuite** &suite)
- void **Image\_xx\_assignment\_to\_self\_xx\_1** (**TestSuite** &suite)
- void **Image\_xx\_type\_convert\_xx\_1** (**TestSuite** &suite)
- void **Image\_xx\_type\_convert\_xx\_2** (**TestSuite** &suite)
- void **Image\_xx\_type\_convert\_xx\_3** (**TestSuite** &suite)

- void **Image\_xx\_attach\_detach\_xx\_1** ([TestSuite](#) &suite)
- void **Image\_xx\_destruct\_xx\_1** ([TestSuite](#) &suite)
- void **Image\_xx\_begin\_end\_xx\_1** ([TestSuite](#) &suite)
- void **Image\_xx\_beginw\_endw\_xx\_1** ([TestSuite](#) &suite)
- void **Image\_xx\_getMean\_xx\_1** ([TestSuite](#) &suite)
- void **Image\_xx\_clear\_xx\_1** ([TestSuite](#) &suite)
- void **Image\_xx\_plus\_eq\_scalar\_xx\_1** ([TestSuite](#) &suite)
- void **Image\_xx\_minus\_eq\_scalar\_xx\_1** ([TestSuite](#) &suite)
- void **Image\_xx\_mul\_eq\_scalar\_xx\_1** ([TestSuite](#) &suite)
- void **Image\_xx\_div\_eq\_scalar\_xx\_1** ([TestSuite](#) &suite)
- void **Image\_xx\_plus\_eq\_array\_xx\_1** ([TestSuite](#) &suite)
- void **Image\_xx\_minus\_eq\_array\_xx\_1** ([TestSuite](#) &suite)
- void **Image\_xx\_mul\_eq\_array\_xx\_1** ([TestSuite](#) &suite)
- void **Image\_xx\_div\_eq\_array\_xx\_1** ([TestSuite](#) &suite)
- void **Image\_xx\_plus\_scalar\_xx\_1** ([TestSuite](#) &suite)
- void **Image\_xx\_minus\_scalar\_xx\_1** ([TestSuite](#) &suite)
- void **Image\_xx\_mul\_scalar\_xx\_1** ([TestSuite](#) &suite)
- void **Image\_xx\_div\_scalar\_xx\_1** ([TestSuite](#) &suite)
- void **Image\_xx\_plus\_array\_xx\_1** ([TestSuite](#) &suite)
- void **Image\_xx\_minus\_array\_xx\_1** ([TestSuite](#) &suite)
- void **Image\_xx\_mul\_array\_xx\_1** ([TestSuite](#) &suite)
- void **Image\_xx\_div\_array\_xx\_1** ([TestSuite](#) &suite)
- void **Image\_xx\_emptyArea\_xx\_1** ([TestSuite](#) &suite)
- void **Image\_xx\_mean\_xx\_1** ([TestSuite](#) &suite)
- void **Image\_xx\_sum\_xx\_1** ([TestSuite](#) &suite)
- void **Image\_xx\_sum\_xx\_2** ([TestSuite](#) &suite)
- void **Image\_xx\_rangeOf\_xx\_1** ([TestSuite](#) &suite)
- void **Image\_xx\_rangeOf\_xx\_2** ([TestSuite](#) &suite)
- void **Image\_xx\_remapRange\_xx\_1** ([TestSuite](#) &suite)
- void **Image\_xx\_squared\_xx\_1** ([TestSuite](#) &suite)
- void **Image\_xx\_toPower\_xx\_1** ([TestSuite](#) &suite)
- void **Image\_xx\_toPower\_xx\_2** ([TestSuite](#) &suite)
- void **Image\_xx\_quadEnergy\_xx\_1** ([TestSuite](#) &suite)
- void **Image\_xx\_overlay\_xx\_1** ([TestSuite](#) &suite)
- void **Image\_xx\_rescale\_xx\_1** ([TestSuite](#) &suite)
- void **Image\_xx\_rescale\_xx\_2** ([TestSuite](#) &suite)
- void **Image\_xx\_rescale\_xx\_3** ([TestSuite](#) &suite)
- void **Image\_xx\_zoomXY\_xx\_1** ([TestSuite](#) &suite)
- void **Image\_xx\_zoomXY\_xx\_2** ([TestSuite](#) &suite)
- void **Image\_xx\_zoomXY\_xx\_3** ([TestSuite](#) &suite)
- void **Image\_xx\_zoomXY\_xx\_4** ([TestSuite](#) &suite)
- void **Image\_xx\_crop\_xx\_1** ([TestSuite](#) &suite)
- void **Image\_xx\_lowPass3x\_xx\_1** ([TestSuite](#) &suite)

- void **Image\_xx\_lowPass3y\_xx\_1** ([TestSuite](#) &suite)
- void **Image\_xx\_lowPass3\_xx\_1** ([TestSuite](#) &suite)
- void **Image\_xx\_lowPass5x\_xx\_1** ([TestSuite](#) &suite)
- void **Image\_xx\_lowPass5y\_xx\_1** ([TestSuite](#) &suite)
- void **Image\_xx\_lowPass5\_xx\_1** ([TestSuite](#) &suite)
- void **Image\_xx\_binomialKernel\_xx\_1** ([TestSuite](#) &suite)
- void **Image\_xx\_sepFiltClean\_xx\_1** ([TestSuite](#) &suite)
- void **Image\_xx\_orientedFilter\_xx\_1** ([TestSuite](#) &suite)
- int **main** (int argc, const char \*\*argv)

### 11.476.1 Detailed Description

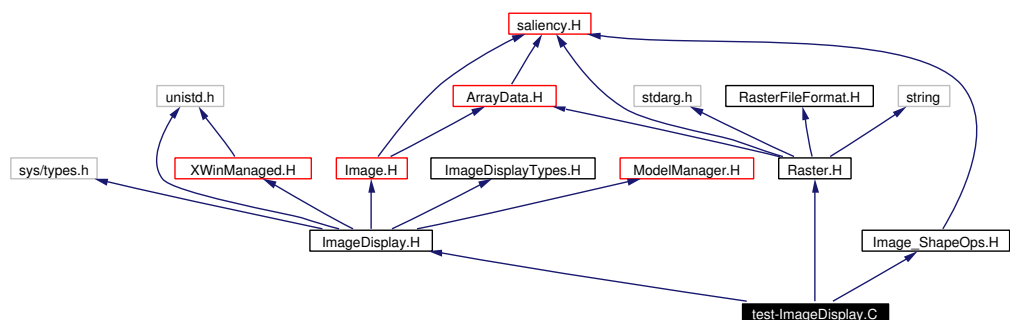
Test [Image](#) class

Definition in file [test-Image.C](#).

## 11.477 test-ImageDisplay.C File Reference

```
#include "ImageDisplay.H"
#include "Image_ShapeOps.H"
#include "Raster.H"
```

Include dependency graph for test-ImageDisplay.C:



### Functions

- `int main (const int argc, const char **argv)`

#### 11.477.1 Detailed Description

Test the [ImageDisplay](#) class

Definition in file [test-ImageDisplay.C](#).

#### 11.477.2 Function Documentation

##### 11.477.2.1 `int main (const int argc, const char ** argv)`

The main method. Try running with various arguments for [ImageDisplay](#) configuration.

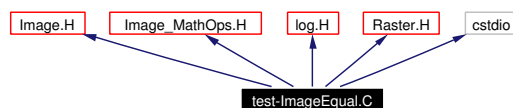
Definition at line 39 of file [test-ImageDisplay.C](#).

References [Image< T >::getHeight\(\)](#), [Image< T >::getWidth\(\)](#), [Image< T >::initialized\(\)](#), [main\(\)](#), [Raster::ReadRGB\(\)](#), and [shiftImage\(\)](#).

## 11.478 test-ImageEqual.C File Reference

```
#include "Image.H"
#include "Image_MathOps.H"
#include "log.H"
#include "Raster.H"
#include <stdio>
```

Include dependency graph for test-ImageEqual.C:



### Functions

- int **main** (int argc, const char \*\*argv)

#### 11.478.1 Detailed Description

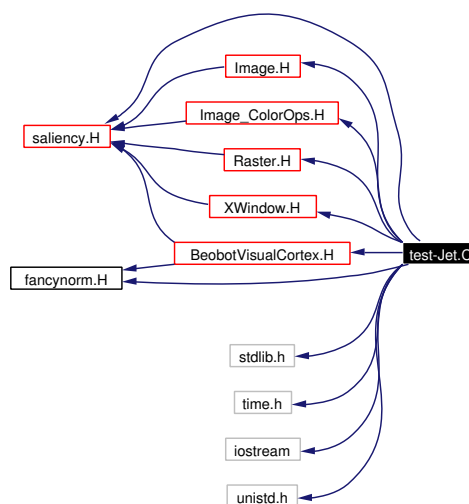
A small program to help testing the [Raster](#) class by loading two images and checking if they are equal.

Definition in file [test-ImageEqual.C](#).

## 11.479 test-Jet.C File Reference

```
#include "saliency.H"  
#include "Image.H"  
#include "BeobotVisualCortex.H"  
#include "Image_ColorOps.H"  
#include "Raster.H"  
#include "fancynorm.H"  
#include <stdlib.h>  
#include <time.h>  
#include "XWindow.H"  
#include <iostream>  
#include <unistd.h>
```

Include dependency graph for test-Jet.C:



## Defines

- #define NBNEIGH 24

## Functions

- int **main** (const int argc, const char \*\*argv)

### 11.479.1 Detailed Description

Test [Jet](#) class

Definition in file [test-Jet.C](#).

## 11.480 test-junction.C File Reference

```
#include "saliency.H"
#include "ModelManager.H"
#include "ModelOptionDefs.H"
#include "Brain.H"
#include "Channels.H"
#include "VisualCortex.H"
#include "SaccadeControllers.H"
#include "SimulationViewerStd.H"
#include "ShapeEstimator.H"
#include "SaliencyMap.H"
#include "TaskRelevanceMap.H"
#include "AttentionGuidanceMap.H"
#include "WinnerTakeAll.H"
#include "Raster.H"
#include "log.H"
#include "FrameSeries.H"
#include <iostream>
#include "SingleChannel.H"
```

Include dependency graph for test-junction.C:

### Functions

- int **main** (const int argc, const char \*\*argv)

#### 11.480.1 Detailed Description

test whether junction detectors work

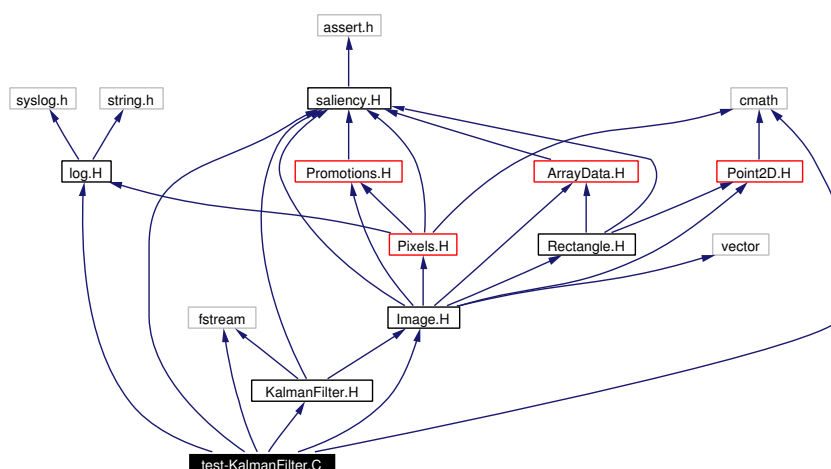
Definition in file [test-junction.C](#).



## 11.481 test-KalmanFilter.C File Reference

```
#include "saliency.H"
#include "log.H"
#include "Image.H"
#include "KalmanFilter.H"
#include <fstream>
#include <cmath>
```

Include dependency graph for test-KalmanFilter.C:



### Functions

- int **main** (const int argc, const char \*\*argv)

### 11.481.1 Detailed Description

- a test program for [KalmanFilter](#).[\[CH\]](#)

Definition in file [test-KalmanFilter.C](#).

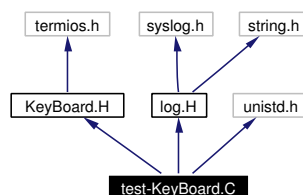
## 11.482 test-KeyBoard.C File Reference

```
#include "KeyBoard.H"
```

```
#include "log.H"
```

```
#include <unistd.h>
```

Include dependency graph for test-KeyBoard.C:



### Functions

- int **main** (const int argc, const char \*\*argv)

### 11.482.1 Detailed Description

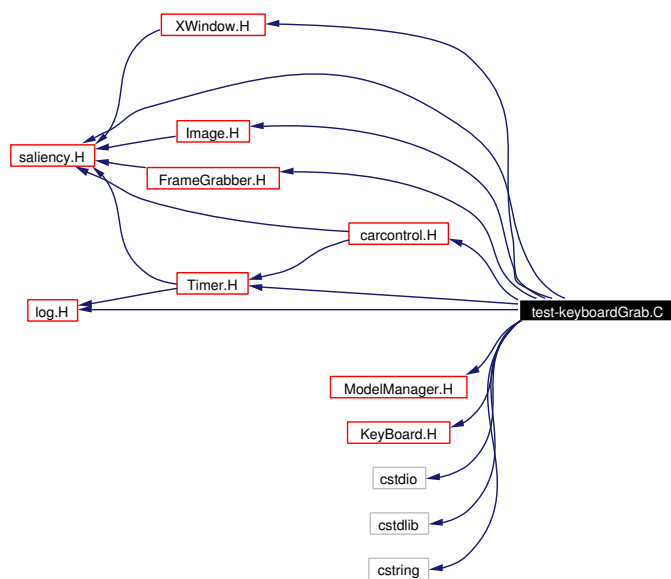
- test the keyboard interface

Definition in file [test-KeyBoard.C](#).

## 11.483 test-keyboardGrab.C File Reference

```
#include "saliency.H"  
#include "Image.H"  
#include "FrameGrabber.H"  
#include "Timer.H"  
#include "XWindow.H"  
#include "log.H"  
#include "ModelManager.H"  
#include "KeyBoard.H"  
#include "carcontrol.H"  
#include <stdio>  
#include <stdlib>  
#include <string>
```

Include dependency graph for test-keyboardGrab.C:



### Defines

- `#define NAVG 20`

## Functions

- int [main](#) (const int argc, const char \*\*argv)

### 11.483.1 Detailed Description

Test keyboard-controlled camera motion

Definition in file [test-keyboardGrab.C](#).

### 11.483.2 Function Documentation

#### 11.483.2.1 int main (const int *argc*, const char \*\* *argv*)

This simple executable tests video frame grabbing through the video4linux driver (see [V4Lgrabber.H](#)) or the IEEE1394 (firewire) grabber (see [IEEE1394grabber.H](#)). Selection of the grabber type is made via the `-fg-type=XX` command-line option.

Definition at line 57 of file `test-keyboardGrab.C`.

References `XWindow::drawImage()`, `Timer::get()`, `SharedPtr< T >::get()`, `Keyboard::getKey()`, `main()`, `NAVG`, `Timer::reset()`, and `uint64`.

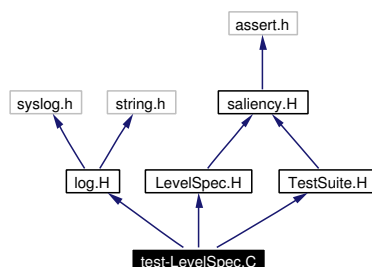
## 11.484 test-LevelSpec.C File Reference

```
#include "log.H"
```

```
#include "LevelSpec.H"
```

```
#include "TestSuite.H"
```

Include dependency graph for test-LevelSpec.C:



### Functions

- void **LevelSpec\_xx\_maxIndex\_xx\_1** ([TestSuite](#) &suite)
- void **LevelSpec\_xx\_maxDepth\_xx\_1** ([TestSuite](#) &suite)
- void **LevelSpec\_xx\_clevOK\_xx\_1** ([TestSuite](#) &suite)
- void **LevelSpec\_xx\_slevOK\_xx\_1** ([TestSuite](#) &suite)
- void **LevelSpec\_xx\_delOK\_xx\_1** ([TestSuite](#) &suite)
- void **LevelSpec\_xx\_csOK\_xx\_1** ([TestSuite](#) &suite)
- void **LevelSpec\_xx\_convert\_xx\_1** ([TestSuite](#) &suite)
- int **main** (int argc, const char \*\*argv)

### 11.484.1 Detailed Description

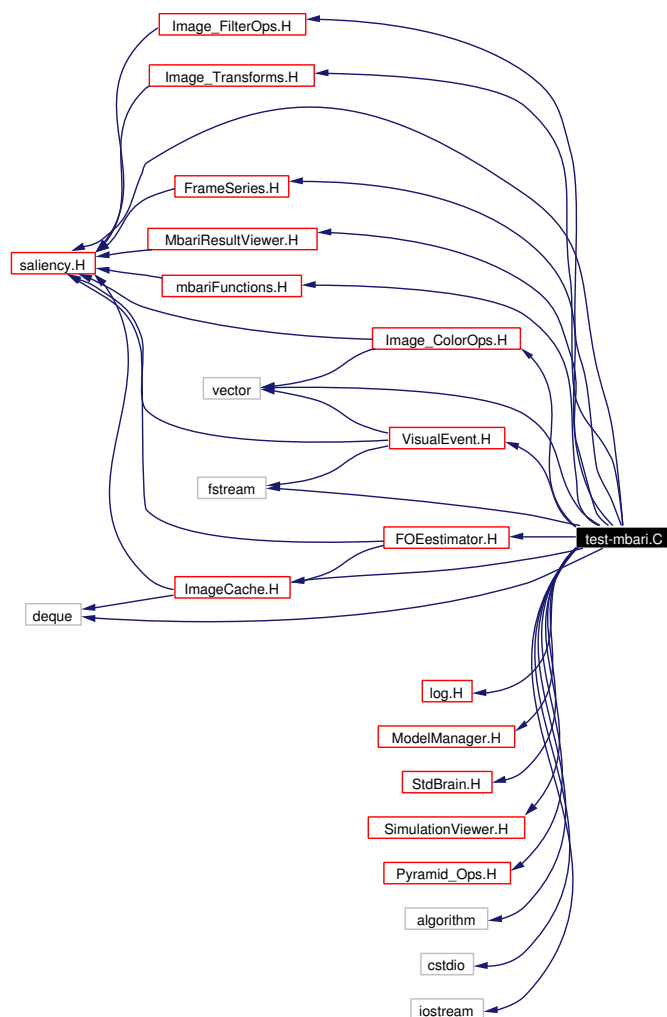
Test the [LevelSpec](#) class

Definition in file [test-LevelSpec.C](#).

## 11.485 test-mbari.C File Reference

```
#include "saliency.H"
#include "log.H"
#include "ModelManager.H"
#include "FrameSeries.H"
#include "VisualEvent.H"
#include "MbariResultViewer.H"
#include "mbariFunctions.H"
#include "ImageCache.H"
#include "Image_FilterOps.H"
#include "Image_ColorOps.H"
#include "Image_Transforms.H"
#include "StdBrain.H"
#include "SimulationViewer.H"
#include "Pyramid_Ops.H"
#include "FOEstimator.H"
#include <fstream>
#include <vector>
#include <deque>
#include <algorithm>
#include <cstdio>
#include <iostream>
```

Include dependency graph for test-mbari.C:



## Functions

- `int main (const int argc, const char **argv)`

### 11.485.1 Detailed Description

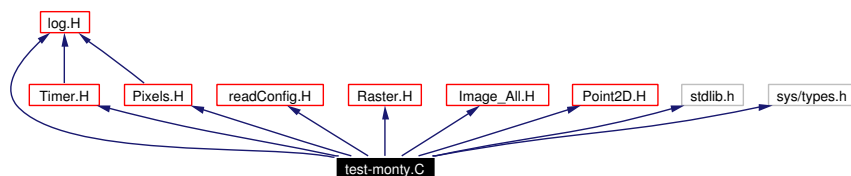
test program to detect marine animals

Definition in file [test-mbari.C](#).

## 11.486 test-monty.C File Reference

```
#include "log.H"  
#include "readConfig.H"  
#include "Raster.H"  
#include "Image_All.H"  
#include "Timer.H"  
#include "Pixels.H"  
#include "Point2D.H"  
#include <stdlib.h>  
#include <sys/types.h>
```

Include dependency graph for test-monty.C:



### Functions

- int **main** (int argc, char \*argv[ ])

### 11.486.1 Detailed Description

Test the monty carlo sparcifier

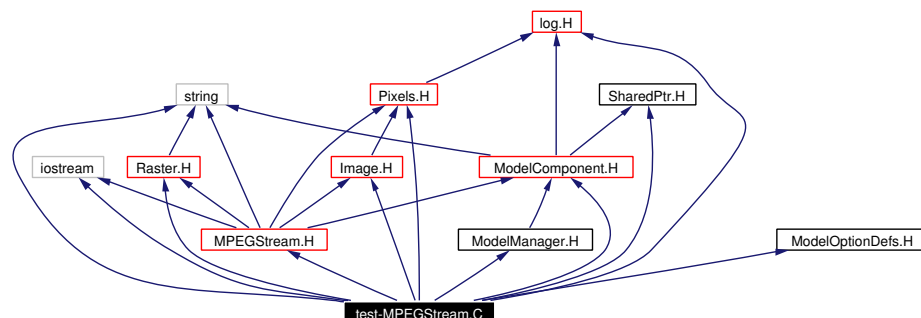
Definition in file [test-monty.C](#).



## 11.487 test-MPEGStream.C File Reference

```
#include <iostream>
#include <string>
#include "MPEGStream.H"
#include "Image.H"
#include "Raster.H"
#include "Pixels.H"
#include "log.H"
#include "SharedPtr.H"
#include "ModelManager.H"
#include "ModelComponent.H"
#include "ModelOptionDefs.H"
```

Include dependency graph for test-MPEGStream.C:



## Functions

- int **main** (const int argc, const char \*\*argv)

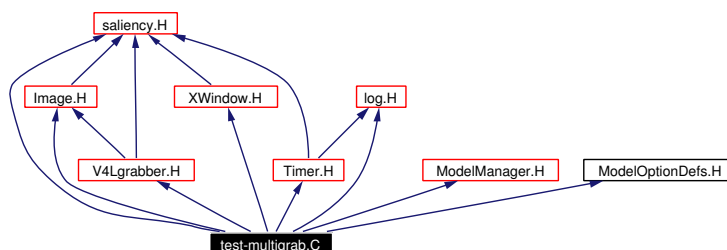
### 11.487.1 Detailed Description

Definition in file [test-MPEGStream.C](#).

## 11.488 test-multigrab.C File Reference

```
#include "saliency.H"
#include "Image.H"
#include "V4Lgrabber.H"
#include "XWindow.H"
#include "log.H"
#include "ModelManager.H"
#include "ModelOptionDefs.H"
#include "Timer.H"
```

Include dependency graph for test-multigrab.C:



### Defines

- `#define NAVG 20`

### Functions

- `int main (const int argc, const char **argv)`

#### 11.488.1 Detailed Description

Test multiple frame grabbing and X display

Definition in file [test-multigrab.C](#).

## 11.488.2 Function Documentation

### 11.488.2.1 `int main (const int argc, const char ** argv)`

This simple executable tests multiple video frame grabbing through the video4linux driver (see [V4Lgrabber.H](#)). For an example with multiple IEEE1394 cameras, see test-stereo.C. If you are grabbing from multiple OV511 USB webcams, you may need to specify the `cams=N` kernel module option, for example with `'options ov511 cams=3'` in your `/etc/modules.conf`.

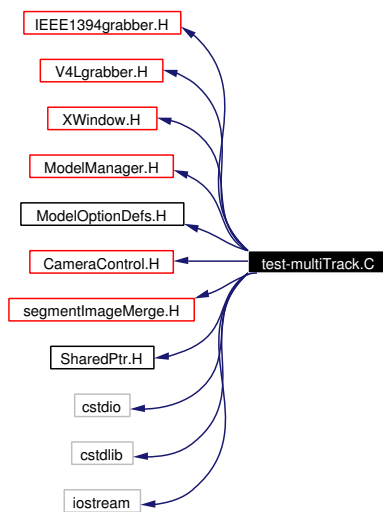
Definition at line 54 of file test-multigrab.C.

References `XWindow::drawImage()`, `Timer::get()`, `Image< T >::getWidth()`, `main()`, `NAVG`, `OPTEXP_NONE`, `Timer::reset()`, and `uint64`.

## 11.489 test-multiTrack.C File Reference

```
#include "IEEE1394grabber.H"  
#include "V4Lgrabber.H"  
#include "XWindow.H"  
#include "ModelManager.H"  
#include "ModelOptionDefs.H"  
#include "CameraControl.H"  
#include "segmentImageMerge.H"  
#include "SharedPtr.H"  
#include <cstdio>  
#include <cstdlib>  
#include <iostream>
```

Include dependency graph for test-multiTrack.C:



## Defines

- `#define NAVG 20`

## Functions

- int **main** (const int argc, const char \*\*argv)

### 11.489.1 Detailed Description

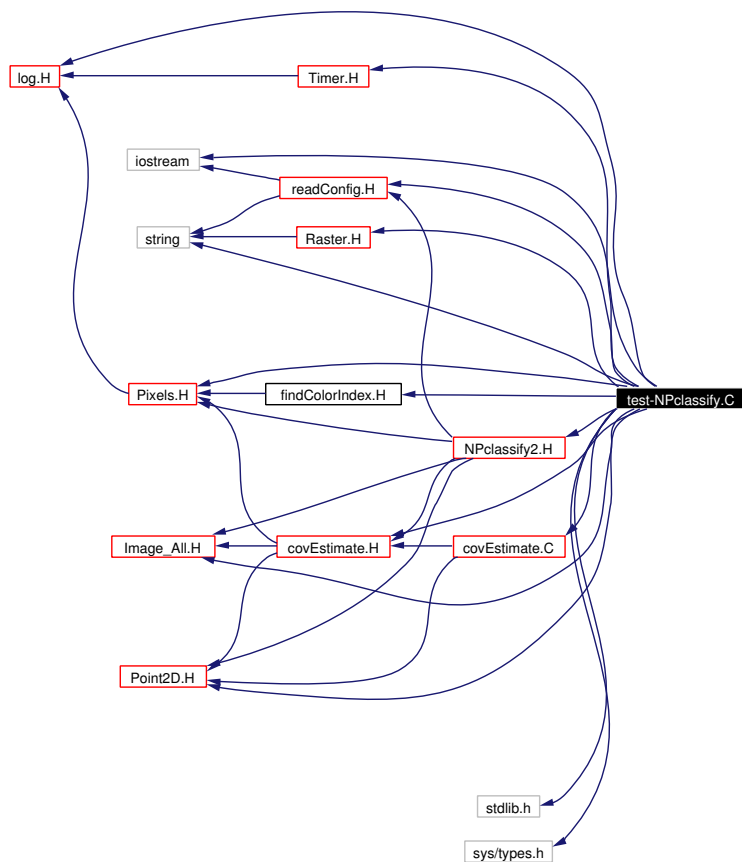
Test IEEE1394 frame grabbing and X display

Definition in file [test-multiTrack.C](#).

## 11.490 test-NPclassify.C File Reference

```
#include "log.H"  
#include "readConfig.H"  
#include "NPclassify2.H"  
#include "covEstimate.H"  
#include "covEstimate.C"  
#include "Raster.H"  
#include "Image_All.H"  
#include "Timer.H"  
#include "Pixels.H"  
#include "Point2D.H"  
#include <stdlib.h>  
#include <iostream>  
#include <string>  
#include <sys/types.h>  
#include "findColorIndex.H"
```

Include dependency graph for test-NPclassify.C:



## Functions

- int [main](#) (int argc, char \*argv[ ])

## Variables

- char \* [configFile](#)  
*This is the configFile name.*
- [readConfig](#) [configIn](#) (25)  
*This is the configFile object.*
- [readConfig](#) [polySet](#) (25)

- [PixRGB](#)< float > [pix](#)  
*generic pixel*
- int [itemNumber](#)  
*number of items if training*

### 11.490.1 Detailed Description

Test the non-parametric classifier

Definition in file [test-NPclassify.C](#).

### 11.490.2 Function Documentation

#### 11.490.2.1 `int main (int argc, char * argv [ ])`

`LINFO("...SIZE %d > %d",NP.NPgetClassSize(i),sortClassSize[j]);`

Definition at line 75 of file `test-NPclassify.C`.

References `configIn`, `Image< T >::drawLine()`, `Timer::get()`, `Image< T >::getHeight()`, `Image< T >::getVal()`, `Image< T >::getWidth()`, `itemNumber`, `main()`, `readConfig::openFile()`, `Raster::ReadGray()`, `Timer::reset()`, `Image< T >::resize()`, `PixRGB< float >::setBlue()`, `PixRGB< float >::setGreen()`, `PixRGB< float >::setRed()`, `Image< T >::setVal()`, `uint64`, `Raster::Visu()`, and `Image< T >::writeText()`.

### 11.490.3 Variable Documentation

#### 11.490.3.1 `char* configFile`

This is the `configFile` name.

Definition at line 66 of file `test-NPclassify.C`.

#### 11.490.3.2 `readConfig configIn(25)`

This is the `configFile` object.

#### 11.490.3.3 `int itemNumber`

number of items if training

Definition at line 73 of file `test-NPclassify.C`.



**11.490.3.4   PixRGB<float> pix**

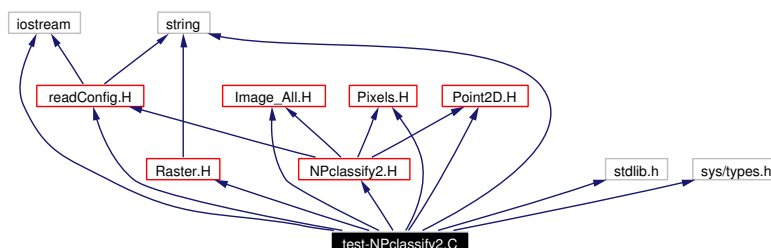
generic pixel

Definition at line 71 of file test-NPclassify.C.

## 11.491 test-NPclassify2.C File Reference

```
#include "readConfig.H"
#include "NPclassify2.H"
#include "Raster.H"
#include "Image_All.H"
#include "Pixels.H"
#include "Point2D.H"
#include <stdlib.h>
#include <iostream>
#include <string>
#include <sys/types.h>
```

Include dependency graph for test-NPclassify2.C:



### Defines

- #define **FEATURES** 36
- #define **DATASIZE** 605

### Functions

- int **main** (int argc, char \*argv[ ])

### Variables

- char \* **configFile**  
*This is the configFile name.*

- [readConfig configIn](#) (25)  
*This is the configFile object.*
- [readConfig polySet](#) (25)
- [PixRGB](#)< float > [pix](#)  
*generic pixel*
- int [itemNumber](#)  
*number of items if training*

### 11.491.1 Detailed Description

Test the non-parametric classifier

Definition in file [test-NPclassify2.C](#).

### 11.491.2 Variable Documentation

#### 11.491.2.1 char\* [configFile](#)

This is the configFile name.

Definition at line 63 of file test-NPclassify2.C.

#### 11.491.2.2 [readConfig configIn](#)(25)

This is the configFile object.

#### 11.491.2.3 int [itemNumber](#)

number of items if training

Definition at line 70 of file test-NPclassify2.C.

#### 11.491.2.4 [PixRGB](#)<float> [pix](#)

generic pixel

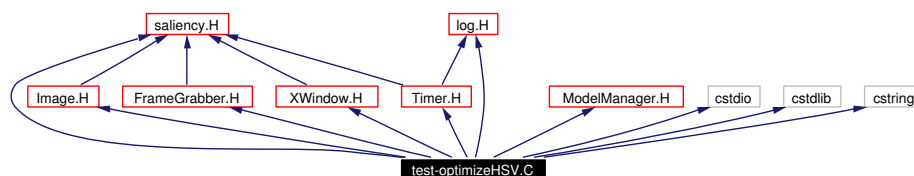
Definition at line 68 of file test-NPclassify2.C.

Referenced by [LPTFoveator::foveate\(\)](#), and [LPTFoveator::getLPT\(\)](#).

## 11.492 test-optimizeHSV.C File Reference

```
#include "saliency.H"
#include "Image.H"
#include "FrameGrabber.H"
#include "Timer.H"
#include "XWindow.H"
#include "log.H"
#include "ModelManager.H"
#include <stdio>
#include <stdlib>
#include <string>
```

Include dependency graph for test-optimizeHSV.C:



### Defines

- #define NAVG 20

### Functions

- int [main](#) (const int argc, const char \*\*argv)

### 11.492.1 Detailed Description

Test optimizing color choice for HSV color segmenter

Definition in file [test-optimizeHSV.C](#).

## 11.492.2 Function Documentation

### 11.492.2.1 `int main (const int argc, const char ** argv)`

This simple executable tests video frame grabbing through the video4linux driver (see [V4Lgrabber.H](#)) or the IEEE1394 (firewire) grabber (see [IEEE1394grabber.H](#)). Selection of the grabber type is made via the `-fg-type=XX` command-line option.

Definition at line 56 of file test-optimizeHSV.C.

References `XWindow::drawImage()`, `Image< PixRGB< byte > >::drawRect()`, `Image< T >::drawRect()`, `SharedPtr< T >::get()`, `Image< PixRGB< byte > >::getHeight()`, `PixRGB< float >::getHSV()`, `Image< PixRGB< byte > >::getVal()`, `Image< PixRGB< byte > >::getWidth()`, `main()`, `NO_INIT`, `Image< T >::resize()`, `PixRGB< float >::setHSV()`, `Image< T >::setVal()`, `sizeX`, and `sizeY`.

## 11.493 test-Pixels.C File Reference

```
#include "log.H"
#include "Pixels.H"
#include "TestSuite.H"
#include <cstdlib>
#include <cstring>
```

Include dependency graph for test-Pixels.C:

### Defines

- #define [REQUIRE\\_PIXEL](#)(pp, rr, gg, bb)  
*Shorthand for checking the RGB values of a pixel:.*

### Functions

- void [Pixels\\_xx\\_constructors\\_xx\\_1](#) ([TestSuite](#) &suite)
- void [Pixels\\_xx\\_conversions\\_xx\\_1](#) ([TestSuite](#) &suite)
- void [Pixels\\_xx\\_operators\\_xx\\_1](#) ([TestSuite](#) &suite)
- void [Pixels\\_xx\\_operators\\_xx\\_2](#) ([TestSuite](#) &suite)
- int [main](#) (int argc, const char \*\*argv)

### 11.493.1 Detailed Description

Test [PixRGB](#) and other Pixels class

Definition in file [test-Pixels.C](#).

### 11.493.2 Define Documentation

#### 11.493.2.1 #define [REQUIRE\\_PIXEL](#)(pp, rr, gg, bb)

**Value:**

```
REQUIRE_EQ(pp.red(), rr); \
REQUIRE_EQ(pp.green(), gg); REQUIRE_EQ(pp.blue(), bb);
```

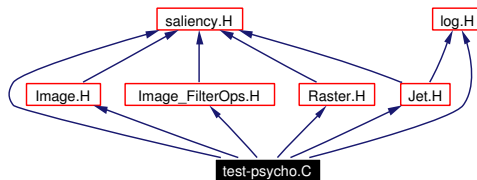
Shorthand for checking the RGB values of a pixel:.

Definition at line 56 of file test-Pixels.C.

## 11.494 test-psycho.C File Reference

```
#include "saliency.H"
#include "log.H"
#include "Image.H"
#include "Image_FilterOps.H"
#include "Raster.H"
#include "Jet.H"
```

Include dependency graph for test-psycho.C:



## Functions

- `int main (int argc, char **argv)`  
*The main routine. Read images, process, display results.*

### 11.494.1 Detailed Description

This is a test of the C++ psychophysics toolkit

Definition in file [test-psycho.C](#).

### 11.494.2 Function Documentation

#### 11.494.2.1 `int main (int argc, char ** argv)`

The main routine. Read images, process, display results.

For command-line arguments, try running the program without any argument.

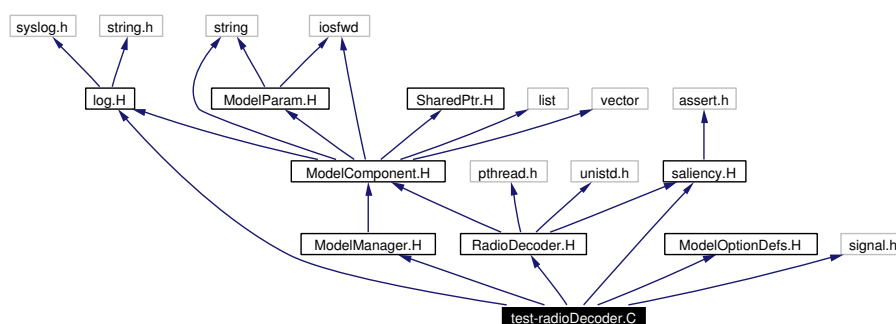
Definition at line 52 of file test-psycho.C.

References `main()`, and `Raster::Visu()`.

## 11.495 test-radioDecoder.C File Reference

```
#include "ModelManager.H"
#include "log.H"
#include "RadioDecoder.H"
#include "saliency.H"
#include "ModelOptionDefs.H"
#include <signal.h>
```

Include dependency graph for test-radioDecoder.C:



### Functions

- void [terminate](#) (int s)  
*Signal handler (e.g., for control-C).*
- int **main** (int argc, const char \*\*argv)

### Variables

- bool [goforever](#) = true  
*Will turn false on interrupt signal.*

### 11.495.1 Detailed Description

Test RC radio decoder

Definition in file [test-radioDecoder.C](#).



## 11.495.2 Function Documentation

### 11.495.2.1 void terminate (int s)

Signal handler (e.g., for control-C).

Definition at line 48 of file test-radioDecoder.C.

References `goforever`.

## 11.495.3 Variable Documentation

### 11.495.3.1 bool `goforever` = true [static]

Will turn false on interrupt signal.

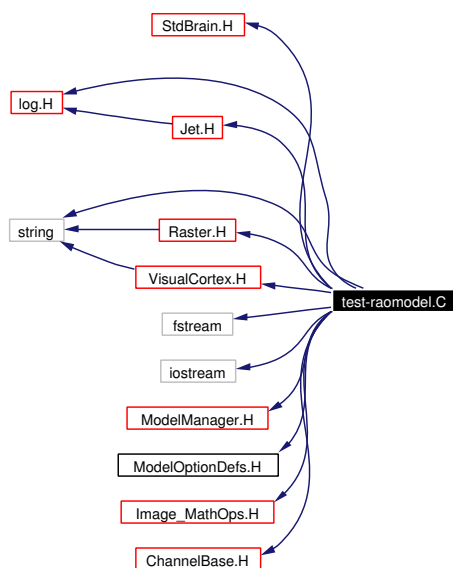
Definition at line 45 of file test-radioDecoder.C.

Referenced by `terminate()`.

## 11.496 test-raomodel.C File Reference

```
#include "StdBrain.H"  
#include "Jet.H"  
#include "log.H"  
#include "Raster.H"  
#include <string>  
#include <fstream>  
#include <iostream>  
#include "ModelManager.H"  
#include "ModelOptionDefs.H"  
#include "VisualCortex.H"  
#include "Image_MathOps.H"  
#include "ChannelBase.H"
```

Include dependency graph for test-raomodel.C:



## Defines

- `#define SMIN 1`

- `#define SMAX 3`

## Functions

- `int main (const int argc, const char **argv)`

### 11.496.1 Detailed Description

test model by Rao et al., Vis Res 2002

Definition in file [test-raomodel.C](#).

## 11.497 test-Raster.C File Reference

```
#include "log.H"
#include "Image.H"
#include "Image_ColorOps.H"
#include "Image_IO.H"
#include "Image_MathOps.H"
#include "Raster.H"
#include "TestSuite.H"
#include <algorithm>
#include <cstdlib>
#include <limits>
#include <unistd.h>
Include dependency graph for test-Raster.C:
```

### Classes

- class **TempFile**
- struct **Randomizer**
- struct **Randomizer**< **PixRGB**< **T** > >

### Functions

- template<class T> **Image**< T > **randImage** (int32 w, int32 h)
- template<class T> **Image**< T > **randImage** ()
- void **Raster\_xx\_img\_gray\_write\_gray\_read\_gray\_xx\_1** (**TestSuite** &suite)
- void **Raster\_xx\_img\_gray\_write\_gray\_read\_rgb\_xx\_1** (**TestSuite** &suite)
- void **Raster\_xx\_img\_gray\_write\_rgb\_read\_gray\_xx\_1** (**TestSuite** &suite)
- void **Raster\_xx\_img\_gray\_write\_rgb\_read\_rgb\_xx\_1** (**TestSuite** &suite)
- void **Raster\_xx\_img\_rgb\_write\_gray\_read\_gray\_xx\_1** (**TestSuite** &suite)
- void **Raster\_xx\_img\_rgb\_write\_gray\_read\_rgb\_xx\_1** (**TestSuite** &suite)
- void **Raster\_xx\_img\_rgb\_write\_rgb\_read\_gray\_xx\_1** (**TestSuite** &suite)
- void **Raster\_xx\_img\_rgb\_write\_rgb\_read\_rgb\_xx\_1** (**TestSuite** &suite)
- int **main** (int argc, const char \*\*argv)

### 11.497.1 Detailed Description

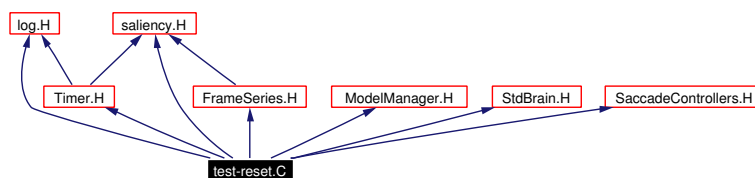
Test [Raster](#) class

Definition in file [test-Raster.C](#).

## 11.498 test-reset.C File Reference

```
#include "log.H"  
#include "saliency.H"  
#include "ModelManager.H"  
#include "StdBrain.H"  
#include "FrameSeries.H"  
#include "SaccadeControllers.H"  
#include "Timer.H"
```

Include dependency graph for test-reset.C:



### Functions

- `int main (const int argc, const char **argv)`

### 11.498.1 Detailed Description

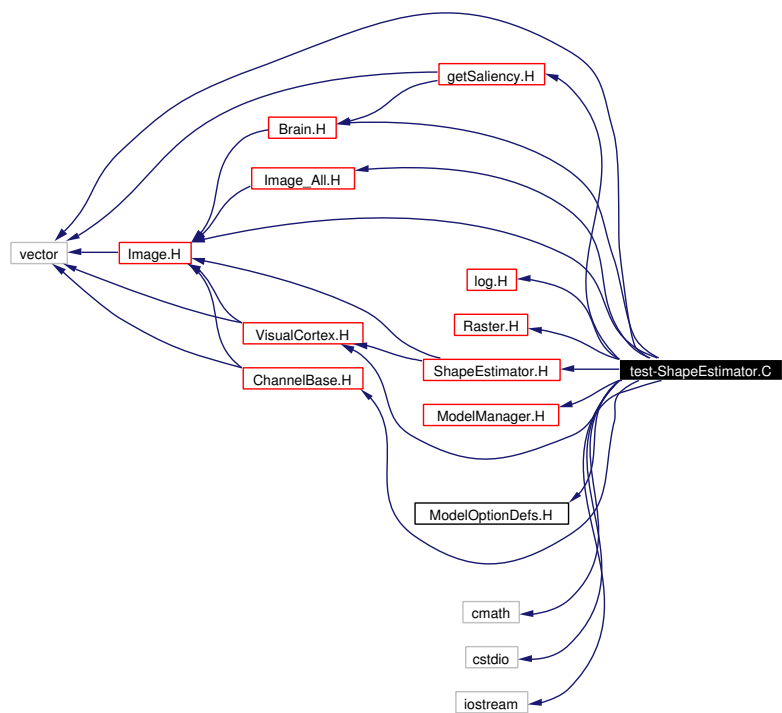
Test the functionality of reset for ModelComponents

Definition in file [test-reset.C](#).

## 11.499 test-ShapeEstimator.C File Reference

```
#include "Brain.H"
#include "getSaliency.H"
#include "Image.H"
#include "Image_All.H"
#include "log.H"
#include "Raster.H"
#include "VisualCortex.H"
#include "ModelManager.H"
#include "ShapeEstimator.H"
#include "ModelOptionDefs.H"
#include "ChannelBase.H"
#include <cmath>
#include <cstdio>
#include <vector>
#include <iostream>
```

Include dependency graph for test-ShapeEstimator.C:



## Defines

- #define [MAX\\_TIME](#) 0.7  
*max time in seconds*

## Functions

- int [main](#) (int argc, const char \*\*argv)  
*The main routine.*

### 11.499.1 Detailed Description

Blackbox tests of SE functionality.

Definition in file [test-ShapeEstimator.C](#).



## 11.499.2 Define Documentation

### 11.499.2.1 `#define MAX_TIME 0.7`

max time in seconds

Definition at line 55 of file test-ShapeEstimator.C.

Referenced by `main()`.

## 11.499.3 Function Documentation

### 11.499.3.1 `int main (int argc, const char ** argv)`

The main routine.

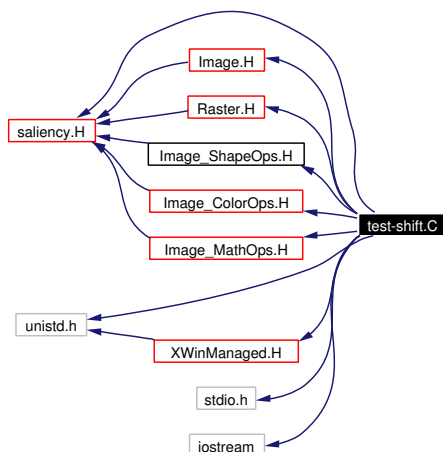
Definition at line 62 of file test-ShapeEstimator.C.

References `Image< T >::getHeight()`, `Image< T >::getWidth()`, `Image< T >::initialized()`, `main()`, `MAX_TIME`, `OPTEXP_CORE`, `Raster::ReadRGB()`, `SEMfeatureMap`, and `VCXNORM_FANCYONE`.

## 11.500 test-shift.C File Reference

```
#include "saliency.H"  
#include "Image.H"  
#include "Raster.H"  
#include "Image_ShapeOps.H"  
#include "Image_ColorOps.H"  
#include "Image_MathOps.H"  
#include "XWinManaged.H"  
#include <stdio.h>  
#include <unistd.h>  
#include <iostream>
```

Include dependency graph for test-shift.C:



### Functions

- `int main (int argc, char **argv)`

### 11.500.1 Detailed Description

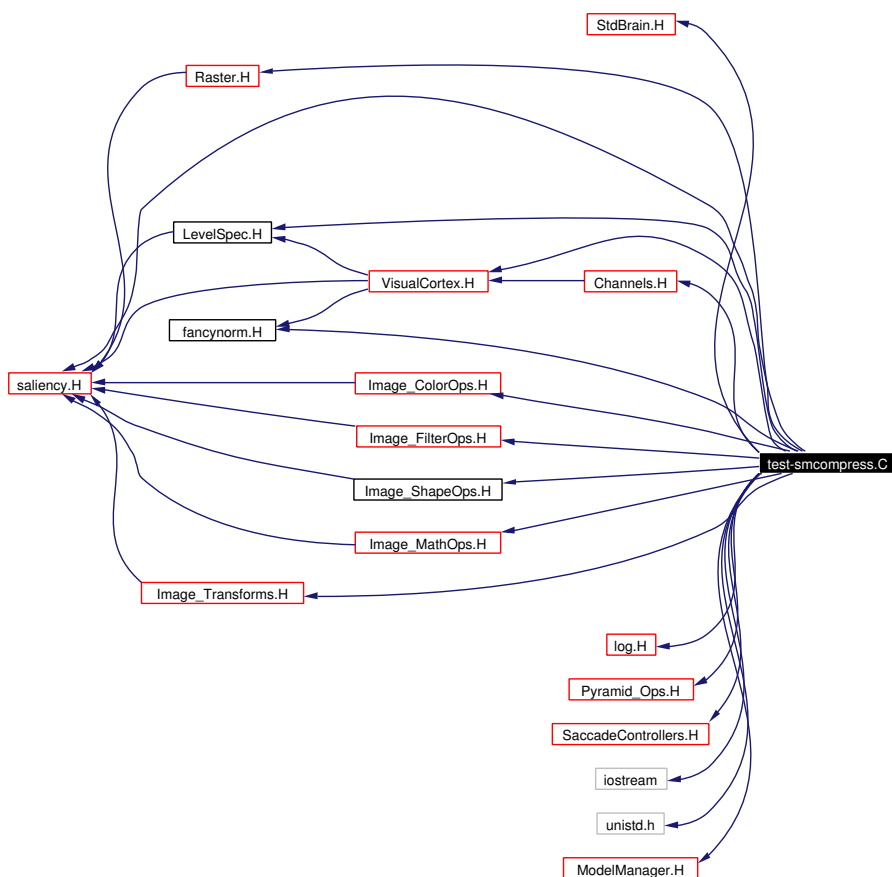
FIXME

Definition in file [test-shift.C](#).

## 11.501 test-smcompress.C File Reference

```
#include "StdBrain.H"
#include "Channels.H"
#include "fancynorm.H"
#include "Image_ColorOps.H"
#include "Image_FilterOps.H"
#include "Image_ShapeOps.H"
#include "Image_MathOps.H"
#include "Image_Transforms.H"
#include "LevelSpec.H"
#include "log.H"
#include "Pyramid_Ops.H"
#include "Raster.H"
#include "saliency.H"
#include "VisualCortex.H"
#include "SaccadeControllers.H"
#include <iostream>
#include <unistd.h>
#include "ModelManager.H"
```

Include dependency graph for test-smcompress.C:



## Defines

- #define **PIXPERDEG** 60  
*Pixels per degree of visual angle (for fixation mask).*
- #define **EYERATE** 120.0f  
*eye tracker sampling rate, in Hz*
- #define **EYESTART** 0.05f  
*start time for us to consider the eye tracking data, in seconds*
- #define **EYEEND** 3.0f  
*end time for us to consider the eye tracking data, in seconds*
- #define **EYEDUR** 2.5f

*duration for which we consider the eye data after 1st fixation, in seconds*

- `#define EYEFWHM 1.0f`  
*full-width at half maximum for eye fixation blobs, in degrees*
- `#define BASELINE 25.0f`  
*baseline for eye fixations (a value of 100 would yield range doubling)*
- `#define MULTIRETINADEPTH 5`  
*depth of multiretina pyramid (a value 5 gives good results):*

## Functions

- `int main (int argc, const char **argv)`  
*The main routine. Read images, process, save results.*

### 11.501.1 Detailed Description

Generate saliency map and YUV files for SMcompress

Definition in file [test-smcompress.C](#).

### 11.501.2 Define Documentation

#### 11.501.2.1 `#define BASELINE 25.0f`

baseline for eye fixations (a value of 100 would yield range doubling)

Definition at line 76 of file test-smcompress.C.

Referenced by `main()`.

#### 11.501.2.2 `#define EYEDUR 2.5f`

duration for which we consider the eye data after 1st fixation, in seconds

Definition at line 70 of file test-smcompress.C.

Referenced by `main()`.

**11.501.2.3 #define EYEEND 3.0f**

end time for us to consider the eye tracking data, in seconds

Definition at line 67 of file test-smcompress.C.

Referenced by main().

**11.501.2.4 #define EYEFWHM 1.0f**

full-width at half maximum for for eye fixation blobs, in degrees

Definition at line 73 of file test-smcompress.C.

Referenced by main().

**11.501.2.5 #define EYERATE 120.0f**

eye tracker sampling rate, in Hz

Definition at line 61 of file test-smcompress.C.

Referenced by main().

**11.501.2.6 #define EYESTART 0.05f**

start time for us to consider the eye tracking data, in seconds

Definition at line 64 of file test-smcompress.C.

Referenced by main().

**11.501.2.7 #define MULTIRETINADEPTH 5**

depth of multiretina pyramid (a value 5 gives good results):

Definition at line 79 of file test-smcompress.C.

Referenced by main().

**11.501.2.8 #define PIXPERDEG 60**

Pixels per degree of visual angle (for fixation mask).

Definition at line 58 of file test-smcompress.C.

Referenced by main().

### 11.501.3 Function Documentation

#### 11.501.3.1 `int main (int argc, const char ** argv)`

The main routine. Read images, process, save results.

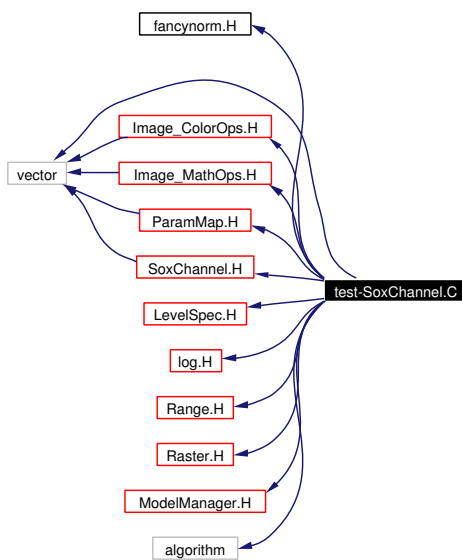
Definition at line 82 of file test-smcompress.C.

References BASELINE, Image< T >::begin(), Image< T >::beginw(), buildPyrGaussian(), byte, Image< T >::clamp(), convGauss(), Image< T >::coordsOk(), Image< T >::countThresh(), decXY(), EYEDUR, EYEEND, EYEFWHM, EYERATE, EYESTART, floodClean(), Image< T >::getDims(), Image< T >::getHeight(), Image< T >::getMinMax(), ImageSet< T >::getPixel(), Image< T >::getSize(), Image< T >::getVal(), Image< T >::getWidth(), getYUV(), intXY(), lowPass9(), Image< T >::lowThresh(), main(), MULTIRETINADEPTH, NO\_INIT, Image< T >::normalize(), Image< T >::paste(), PIXPERDEG, quickLocalAvg(), Raster::ReadRGB(), rescale(), Image< T >::setVal(), sqrt(), Raster::WriteGray(), Raster::WriteRGB(), and ZEROS.

## 11.502 test-SoxChannel.C File Reference

```
#include "fancynorm.H"  
#include "Image_ColorOps.H"  
#include "Image_MathOps.H"  
#include "LevelSpec.H"  
#include "log.H"  
#include "ParamMap.H"  
#include "Range.H"  
#include "Raster.H"  
#include "SoxChannel.H"  
#include "ModelManager.H"  
#include <algorithm>  
#include <vector>
```

Include dependency graph for test-SoxChannel.C:



## Functions

- int **main** (const int argc, const char \*\*argv)



### **11.502.1 Detailed Description**

Test the [SoxChannel](#) class

Definition in file [test-SoxChannel.C](#).

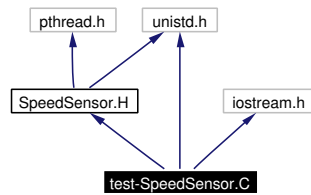
## 11.503 test-SpeedSensor.C File Reference

```
#include "SpeedSensor.H"
```

```
#include <iostream.h>
```

```
#include <unistd.h>
```

Include dependency graph for test-SpeedSensor.C:



### Functions

- `int main ()`

#### 11.503.1 Detailed Description

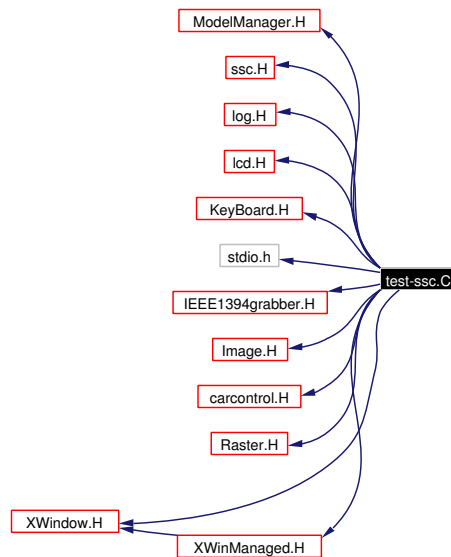
Test the [SpeedSensor](#) class

Definition in file [test-SpeedSensor.C](#).

## 11.504 test-ssc.C File Reference

```
#include "ModelManager.H"
#include "ssc.H"
#include "log.H"
#include "lcd.H"
#include "KeyBoard.H"
#include <stdio.h>
#include "IEEE1394grabber.H"
#include "Image.H"
#include "carcontrol.H"
#include "Raster.H"
#include "XWindow.H"
#include "XWinManaged.H"
```

Include dependency graph for test-ssc.C:



## Functions

- int **main** (int argc, const char \*\*argv)

### 11.504.1 Detailed Description

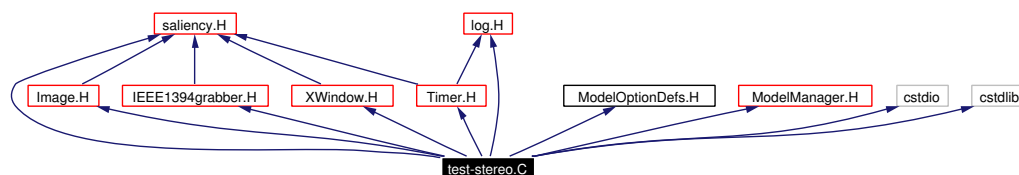
Test [SSC](#) serial servo controller

Definition in file [test-ssc.C](#).

## 11.505 test-stereo.C File Reference

```
#include "saliency.H"
#include "Image.H"
#include "IEEE1394grabber.H"
#include "Timer.H"
#include "XWindow.H"
#include "log.H"
#include "ModelOptionDefs.H"
#include "ModelManager.H"
#include <stdio>
#include <stdlib>
```

Include dependency graph for test-stereo.C:



### Defines

- `#define NAVG 20`

### Functions

- `Image< PixRGB< byte > > ImageRotate (Image< PixRGB< byte > > img)`
- `int main (const int argc, const char **argv)`

### 11.505.1 Detailed Description

Test stereovision algorithm

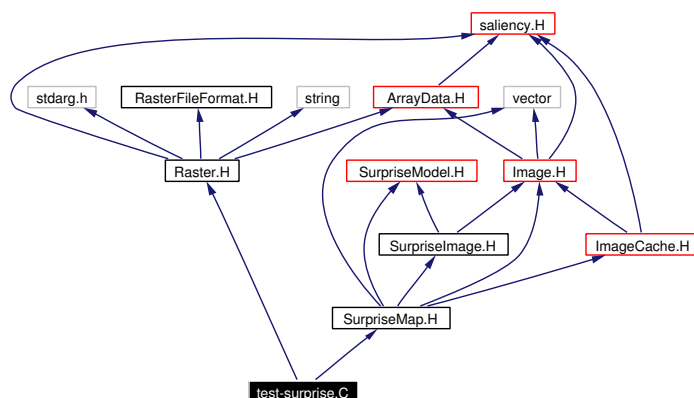
Definition in file [test-stereo.C](#).

## 11.506 test-surprise.C File Reference

```
#include "Raster.H"
```

```
#include "SurpriseMap.H"
```

Include dependency graph for test-surprise.C:



### Defines

- `#define QLEN 3`
- `#define UPDFAC 0.75`
- `#define NUPDFAC 0.75`
- `#define INIVAL 0.0`
- `#define VARIANCE 25.0`
- `#define NEIGHSIGMA 0.5f`
- `#define LOCSIGMA 3.0f`
- `#define NITER 20`

### Functions

- `int main (const int argc, const char **argv)`

#### 11.506.1 Detailed Description

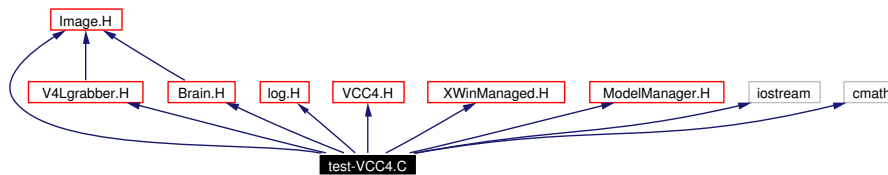
test basic behavior of [SurpriseMap](#) and contents

Definition in file [test-surprise.C](#).

## 11.507 test-VCC4.C File Reference

```
#include "Image.H"
#include "log.H"
#include "V4Lgrabber.H"
#include "VCC4.H"
#include "Brain.H"
#include "XWinManaged.H"
#include "ModelManager.H"
#include <iostream>
#include <cmath>
```

Include dependency graph for test-VCC4.C:



## Defines

- `#define CAM_WIDTH 320`
- `#define CAM_HEIGHT 240`
- `#define CAM_DISPLAY NULL`
- `#define CAM_WINDOW_NAME "Saliency Calculated"`
- `#define CAM_PREV_NAME "Captured Image"`
- `#define CAM_VIDEO_DEVICE "/dev/video0"`
- `#define CAM_VIDEO_CHANNEL 2`
- `#define CAM_SERIAL_DEVICE "/dev/ttyS0"`
- `#define VCC4_HDEG 50`
- `#define VCC4_WDEG 67`
- `#define MAXPAN ((float)100.0)`
- `#define MINPAN ((float)-100.0)`
- `#define MAXTILT ((float)30.0)`
- `#define MINTILT ((float)-30.0)`
- `#define TOO_MUCH_TIME 0.7`

## Functions

- float **sqr** (float x)
- float **randang** ()
- int **main** (const int argc, const char \*\*argv)

### 11.507.1 Detailed Description

Use a pan-tilt camera for executing saccades

Definition in file [test-VCC4.C](#).

### 11.507.2 Function Documentation

#### 11.507.2.1 int main (const int *argc*, const char \*\* *argv*)

This program takes input from a camera, calculates the most salient spot(s) and moves the camera there according to the following rules:

1) take image from camera 2) calculate next most salient spot 3) If (spot is already in list) goto 2 4) store spot in list 5) move to the spot 6) goto 1

The list is finite (small) sized and "forgets" the oldest entries. When the search for the next most salient spot in a given image takes too much time, the camera is moved by a random amount.

Definition at line 98 of file test-VCC4.C.

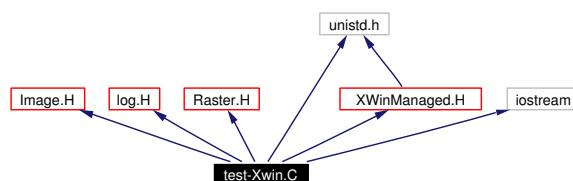
References [main\(\)](#), and [MYLOGVERB](#).



## 11.508 test-Xwin.C File Reference

```
#include "Image.H"  
#include "log.H"  
#include "Raster.H"  
#include "XWinManaged.H"  
#include <unistd.h>  
#include <iostream>
```

Include dependency graph for test-Xwin.C:



### Functions

- int **main** (int argc, char \*\*argv)

### 11.508.1 Detailed Description

Test the Xwindow class

Definition in file [test-Xwin.C](#).

## 11.509 testimagesat.C File Reference

```
#include <fstream>
```

```
#include "FrameSeries.H"
```

Include dependency graph for testimagesat.C:

### Functions

- `int main (const int argc, const char **argv)`

#### 11.509.1 Detailed Description

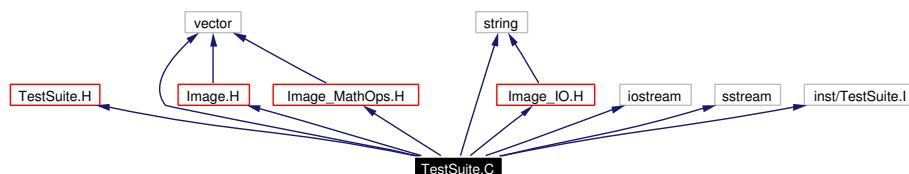
show saccades locations on satellite pictures. Usage : "testimage rad image" where rad is an int and image the common part of the picture and its scanpaths filenames (without extension).

Definition in file [testimagesat.C](#).

## 11.510 TestSuite.C File Reference

```
#include "TestSuite.H"
#include "Image.H"
#include "Image_IO.H"
#include "Image_MathOps.H"
#include <iostream>
#include <string>
#include <sstream>
#include <vector>
#include "inst/TestSuite.I"
```

Include dependency graph for TestSuite.C:



### Classes

- struct **TestData**
- struct **TestSuite::Impl**

### Defines

- #define **INST\_CLASS** TestSuite::

#### 11.510.1 Detailed Description

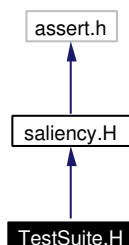
Class to manage a suite of tests

Definition in file [TestSuite.C](#).

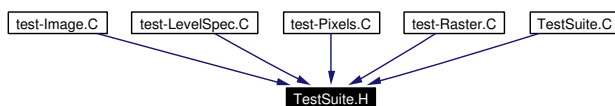
## 11.511 TestSuite.H File Reference

```
#include "saliency.H"
```

Include dependency graph for TestSuite.H:



This graph shows which files directly or indirectly include this file:



## Classes

- class [TestSuite](#)

*TestSuite manages a set of tests.*

## Defines

- #define **ADD\_TEST**(x) addTest(#x, &x);
- #define **REQUIRE\_EQ**(expr, targ) suite.requireEq((expr), targ, \_\_LINE\_\_, #expr)
- #define **REQUIRE\_EQFP**(expr, targ, prec) suite.requireEq((expr), targ, prec, \_\_LINE\_\_, #expr)
- #define **REQUIRE**(expr) suite.require((expr), \_\_LINE\_\_, #expr)
- #define **REQUIRE\_NEQ**(lhs, rhs) suite.require((lhs), TestSuite::NEQ, (rhs), \_\_LINE\_\_, #lhs, #rhs)
- #define **REQUIRE\_LT**(lhs, rhs) suite.require((lhs), TestSuite::LT, (rhs), \_\_LINE\_\_, #lhs, #rhs)

- #define **REQUIRE\_LTE**(lhs, rhs) suite.require((lhs), TestSuite::LTE, (rhs), \_\_LINE\_\_, #lhs, #rhs)
- #define **REQUIRE\_GT**(lhs, rhs) suite.require((lhs), TestSuite::GT, (rhs), \_\_LINE\_\_, #lhs, #rhs)
- #define **REQUIRE\_GTE**(lhs, rhs) suite.require((lhs), TestSuite::GTE, (rhs), \_\_LINE\_\_, #lhs, #rhs)

## Typedefs

- typedef void(\* **TestFunc** )(TestSuite &)

### 11.511.1 Detailed Description

Class to manage a suite of tests

Definition in file [TestSuite.H](#).

## 11.512 Timer.H File Reference

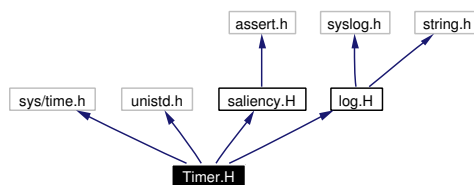
```
#include <sys/time.h>
```

```
#include <unistd.h>
```

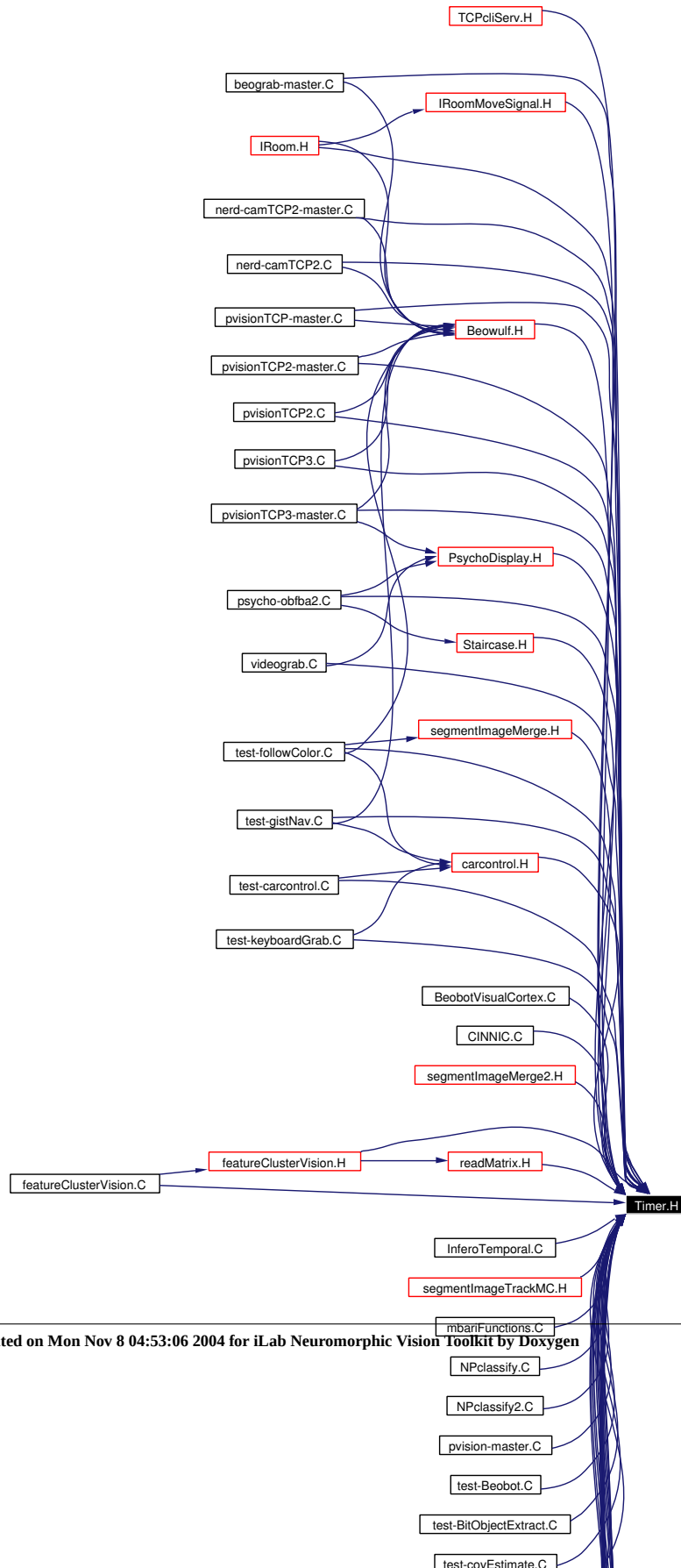
```
#include "saliency.H"
```

```
#include "log.H"
```

Include dependency graph for Timer.H:



This graph shows which files directly or indirectly include this file:



## Classes

- class [Timer](#)

*This class implements a real-time timer.*

### 11.512.1 Detailed Description

A simple class for precise time measurements

Definition in file [Timer.H](#).



## 11.513 train-gist\_headSize.C File Reference

```
#include "Raster.H"
#include "XWindow.H"
#include "XWinManaged.H"
#include "saliency.H"
#include "ModelManager.H"
#include "ComplexChannel.H"
#include "Channels.H"
#include "VisualCortex.H"
#include "FrameSeries.H"
#include "ModelOptionDefs.H"
#include "StdBrain.H"
#include "Jet.H"
#include "ShapeEstimator.H"
#include "Image.H"
#include "Pyramid_Ops.H"
#include "Image_MathOps.H"
#include "Image_ShapeOps.H"
#include "Image_FilterOps.H"
#include "Image_ColorOps.H"
#include "Image_Transforms.H"
#include "ImageSet.H"
#include "PyrBuilder.H"
#include "Point2D.H"
#include "Dims.H"
#include <cstdio>
#include <cstdlib>
#include <cstring>
#include <stdio.h>
#include <iostream>
```

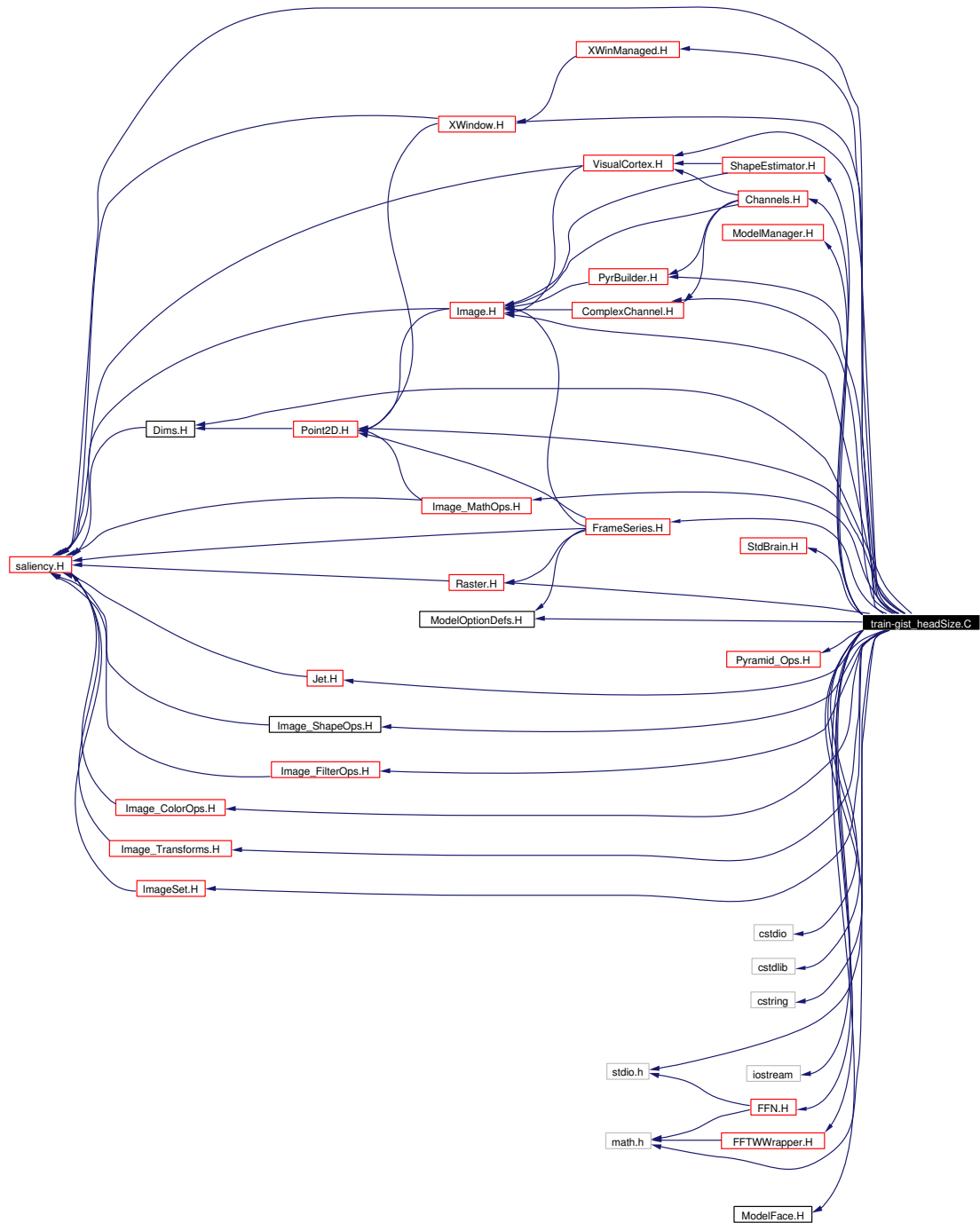
```
#include <math.h>
```

```
#include "FFN.H"
```

```
#include "FTWrapper.H"
```

```
#include "ModelFace.H"
```

Include dependency graph for train-gist\_headSize.C:



## Defines

- #define [LEARN\\_RATE](#) 0.5

## Functions

- int **setupTrainCases** ()
- void **train** (int nSamples)
- int **setupTestCases** ()
- void **test** (int nSamples)
- void **setupFFTW** ([Image](#)< float > img)
- int **getPrimeLev** ([Image](#)< float > img)
- void **setupGaborMask** ([Image](#)< float > img)
- [Image](#)< float > **normalize** ([Image](#)< float > img)
- [Image](#)< float > **getFftImage** (double \*\*res, int w, int h)
- void **fftCompute** ([Image](#)< float > img, double \*\*fftOut)
- void **getFeatureVector** (double \*\*res, double \*vec, int w, int h)
- void **freeFftwData** ([Image](#)< float > img)
- void **printRegion** ([Image](#)< float > img, int sX, int eX, int dX, int sY, int eY, int dY)
- int **main** (const int argc, const char \*\*argv)

## Variables

- [CloseButtonListener](#) **wList**
- FFTWWrapper \* **fftw**
- [FeedForwardNetwork](#) \* **ffn\_pl**
- double \* **input** = NULL
- double \* **iptr** = NULL
- double \*\* **outFftw** = NULL
- double \*\*\*\* **gaborMask** = NULL
- [Image](#)< float > **gaborMaskImg** [NUM\_G\_LEV][NUM\_G\_DIR]
- double \*\* **in** = NULL
- double \*\* **out** = NULL
- double \*\* **inTest** = NULL
- double \*\* **outTest** = NULL

### 11.513.1 Detailed Description

train a ffn to recognize head size using fourier transform of the image

Definition in file [train-gist\\_headSize.C](#).

## 11.513.2 Define Documentation

### 11.513.2.1 `#define LEARN_RATE 0.5`

head-size-recognizer training program

Definition at line 79 of file train-gist\_headSize.C.

## 11.514 train-gist\_headSize2.C File Reference

```
#include "Raster.H"
#include "XWindow.H"
#include "XWinManaged.H"
#include "saliency.H"
#include "ModelManager.H"
#include "ComplexChannel.H"
#include "Channels.H"
#include "VisualCortex.H"
#include "FrameSeries.H"
#include "ModelOptionDefs.H"
#include "StdBrain.H"
#include "Jet.H"
#include "ShapeEstimator.H"
#include "Image.H"
#include "Pyramid_Ops.H"
#include "Image_MathOps.H"
#include "Image_ShapeOps.H"
#include "Image_FilterOps.H"
#include "Image_ColorOps.H"
#include "Image_Transforms.H"
#include "ImageSet.H"
#include "PyrBuilder.H"
#include "Point2D.H"
#include "Dims.H"
#include <cstdio>
#include <cstdlib>
#include <cstring>
#include <stdio.h>
#include <iostream>
```

```
#include <math.h>
```

```
#include "FFN.H"
```

```
#include "ModelFace.H"
```

Include dependency graph for train-gist\_headSize2.C:

## Defines

- #define [LEARN\\_RATE](#) 0.5
- #define [I\\_WIDTH](#) 320
- #define [I\\_HEIGHT](#) 240

## Functions

- int **setupTrainCases** ([SharedPtr](#)< [StdBrain](#) > brain)
- void **train** (int nSamples)
- int **setupTestCases** ([SharedPtr](#)< [StdBrain](#) > brain)
- void **test** (int nSamples)
- void **getFeatureVector** ([SharedPtr](#)< [StdBrain](#) > brain, double \*vec, int w, int h)
- [Image](#)< float > **normalize** ([Image](#)< float > img)
- void **printRegion** ([Image](#)< float > img, int sX, int eX, int dX, int sY, int eY, int dY)
- int **main** (const int argc, const char \*\*argv)

## Variables

- [CloseButtonListener](#) **wList**
- [FeedForwardNetwork](#) \* **ffn\_pl2**
- double \*\* **in** = NULL
- double \*\* **out** = NULL
- double \*\* **inTest** = NULL
- double \*\* **outTest** = NULL

### 11.514.1 Detailed Description

train a ffn to recognize head size using available features of the image CAN ONLY WORK WITH A SPECIFIED IMAGE SIZE

Definition in file [train-gist\\_headSize2.C](#).

## 11.514.2 Define Documentation

### 11.514.2.1 `#define LEARN_RATE 0.5`

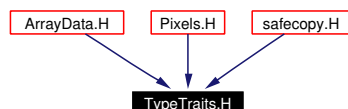
head-size-recognizer training program

Definition at line 80 of file train-gist\_headSize2.C.



## 11.515 TypeTraits.H File Reference

This graph shows which files directly or indirectly include this file:



### Classes

- struct [TypeTraits](#)  
*The generic TypeTraits implementation.*
- struct **TypeTraits**< **bool** >
- struct **TypeTraits**< **char** >
- struct **TypeTraits**< **unsigned char** >
- struct **TypeTraits**< **short** >
- struct **TypeTraits**< **unsigned short** >
- struct **TypeTraits**< **int** >
- struct **TypeTraits**< **unsigned int** >
- struct **TypeTraits**< **long** >
- struct **TypeTraits**< **unsigned long** >
- struct **TypeTraits**< **float** >
- struct **TypeTraits**< **double** >
- struct **TypeTraits**< **long double** >

### 11.515.1 Detailed Description

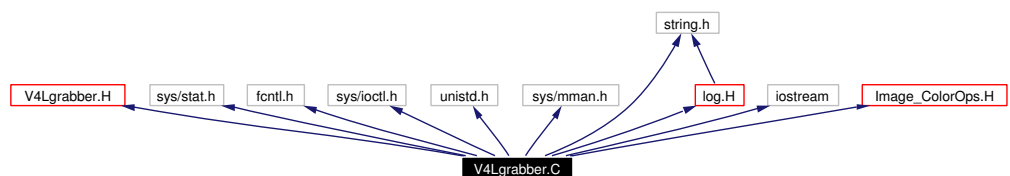
Template approach to getting compile-time information about types.

Definition in file [TypeTraits.H](#).

## 11.516 V4Lgrabber.C File Reference

```
#include "V4Lgrabber.H"
#include <sys/stat.h>
#include <fcntl.h>
#include <sys/ioctl.h>
#include <unistd.h>
#include <sys/mman.h>
#include <string.h>
#include <iostream>
#include "log.H"
#include "Image_ColorOps.H"
```

Include dependency graph for V4Lgrabber.C:



### Defines

- `#define MYLOGID fd`  
for id-logging; see [log.H](#):

### Variables

- [V4LPalette palettes](#) [12]

### 11.516.1 Detailed Description

Interface with a video4linux frame grabber

Definition in file [V4Lgrabber.C](#).

## 11.516.2 Define Documentation

### 11.516.2.1 #define MYLOGID fd

for id-logging; see [log.H](#):

Definition at line 49 of file V4Lgrabber.C.

## 11.516.3 Variable Documentation

### 11.516.3.1 struct [V4LPalette](#) palettes[12]

**Initial value:**

```
{
  { VIDEO_PALETTE_RGB24,      3,   24, "bgr24",  GMRGB24 },
  { VIDEO_PALETTE_RGB24 | 0x80, 3,   24, "rgb24",  GMRGB24 },
  { VIDEO_PALETTE_RGB32,      4,   32, "bgr32",  GMRGB32 },
  { VIDEO_PALETTE_RGB32 | 0x80, 4,   32, "rgb32",  GMRGB32 },
  { VIDEO_PALETTE_YUYV,       2,   24, "yuyv 16bpp", GMYUYV },
  { VIDEO_PALETTE_YUV422,     2,   24, "yuv422 16bpp", GMYUV422 },
  { VIDEO_PALETTE_RGB565,     2,   16, "rgb565",  GMRGB565 },
  { VIDEO_PALETTE_RGB555,     2,   15, "rgb555",  GMRGB555 },
  { VIDEO_PALETTE_YUV420,     1.5, 24, "yuv420 12bpp", GMYUV420 },
  { VIDEO_PALETTE_YUV420P,    1.5, 24, "yuv420 planar 12bpp", GMYUV420P },
  { VIDEO_PALETTE_GREY,       1,    8, "grayscale 8bpp", GMGREY },
  { -1                        }
}
```

array of all known palettes high preference modes have lower indices

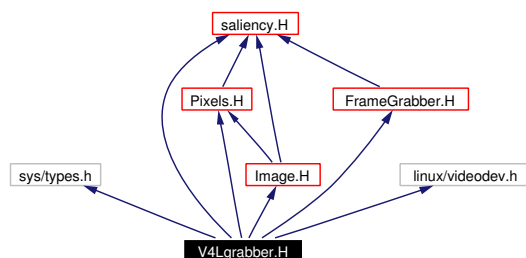
Definition at line 53 of file V4Lgrabber.C.

Referenced by `V4Lgrabber::start1()`.

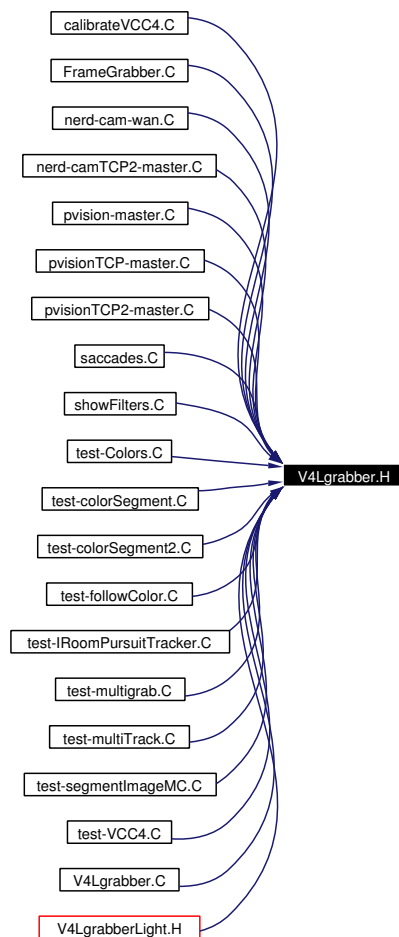
## 11.517 V4Lgrabber.H File Reference

```
#include <sys/types.h>
#include "saliency.H"
#include "Pixels.H"
#include "Image.H"
#include <linux/videodev.h>
#include "FrameGrabber.H"
```

Include dependency graph for V4Lgrabber.H:



This graph shows which files directly or indirectly include this file:



## Classes

- class [V4Lgrabber](#)  
*Definition and access functions for video4linux frame grabber.*
- struct [V4LPalette](#)  
*A V4L palette entry.*

## Typedefs

- typedef long long int **\_\_s64**
- typedef long long unsigned int **\_\_u64**

### 11.517.1 Detailed Description

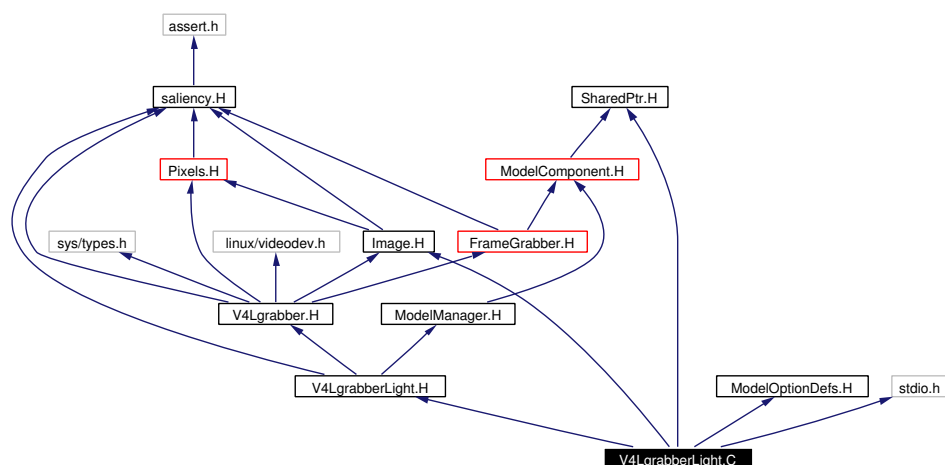
Definition and access functions for video4linux grabber

Definition in file [V4Lgrabber.H](#).

## 11.518 V4LgrabberLight.C File Reference

```
#include "V4LgrabberLight.H"
#include "SharedPtr.H"
#include "Image.H"
#include "ModelOptionDefs.H"
#include <stdio.h>
```

Include dependency graph for V4LgrabberLight.C:



### 11.518.1 Detailed Description

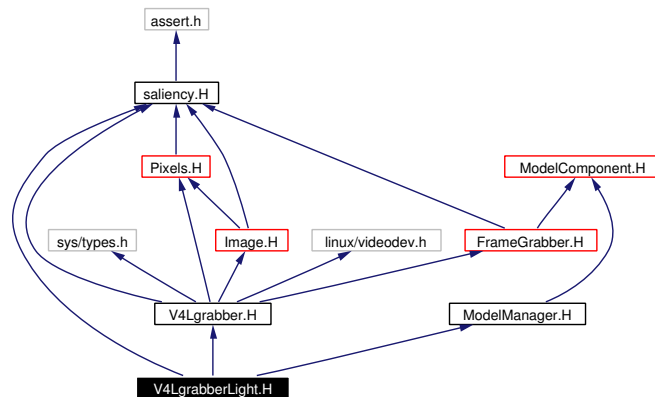
a wrapper class for [V4Lgrabber](#) that provides an easy interface without having to worry about [ModelManager](#) & Co

Definition in file [V4LgrabberLight.C](#).

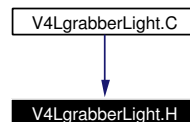
## 11.519 V4LgrabberLight.H File Reference

```
#include "saliency.H"
#include "V4Lgrabber.H"
#include "ModelManager.H"
```

Include dependency graph for V4LgrabberLight.H:



This graph shows which files directly or indirectly include this file:



### Classes

- class [V4LgrabberLight](#)  
Wrapper class for [V4Lgrabber](#) that restores the original interface.

### 11.519.1 Detailed Description

a wrapper class for [V4Lgrabber](#) that provides an easy interface without having to worry about [ModelManager](#) & Co

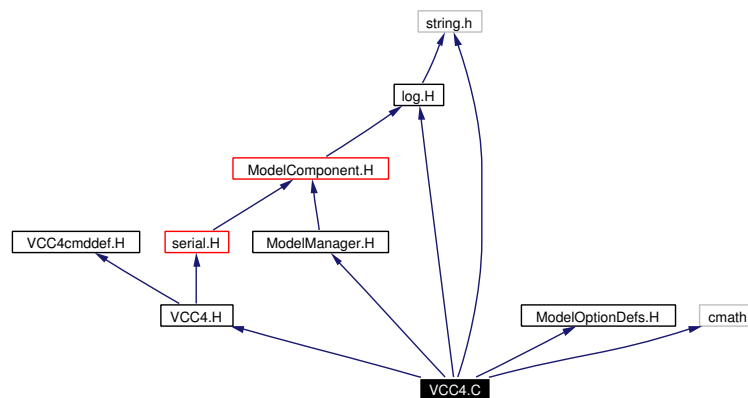
Definition in file [V4LgrabberLight.H](#).



## 11.520 VCC4.C File Reference

```
#include "VCC4.H"  
#include "log.H"  
#include "ModelManager.H"  
#include "ModelOptionDefs.H"  
#include <cmath>  
#include <string.h>
```

Include dependency graph for VCC4.C:



### 11.520.1 Detailed Description

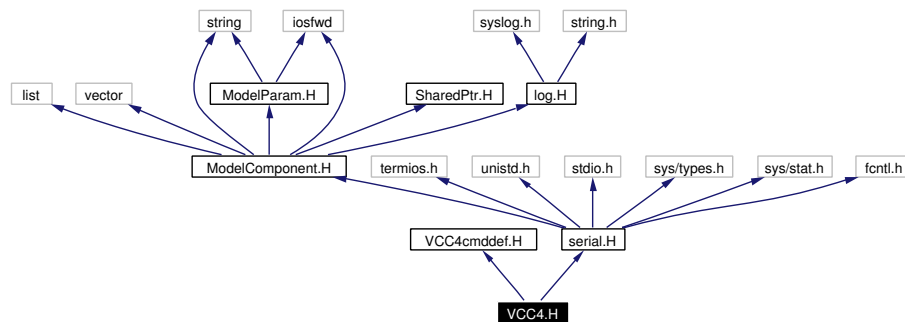
class for interfacing with the pan-tilt mechanism of a  
Definition in file [VCC4.C](#).

## 11.521 VCC4.H File Reference

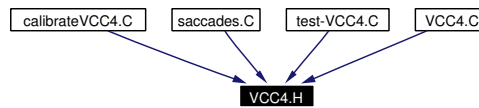
```
#include "VCC4cmddef.H"
```

```
#include "serial.H"
```

Include dependency graph for VCC4.H:



This graph shows which files directly or indirectly include this file:



## Classes

- class [VCC4](#)  
*interface with the pan-tilt mechanism of a Canon VC-C4 camera*

## Defines

- #define **VCC4\_EMBED\_NONE** 0
- #define **VCC4\_EMBED\_NUM** 1
- #define **VCC4\_EMBED\_ANG** 2
- #define **VCC4\_EMBED\_STR** 3
- #define **VCC4\_SUCCESS** 0
- #define **VCC4\_OPEN\_ERR** 1
- #define **VCC4\_INVALID\_COMMAND** 2

- `#define VCC4_INVALID_PARAM 3`
- `#define VCC4_IO_ERR 4`
- `#define VCC4_REPLY_ERR 5`
- `#define VCC4_NO_REPLY 6`
- `#define VCC4_INVALID_NUMBER 7`
- `#define VCC4_INVALID_ANGLE 8`
- `#define VCC4_ERR_MASK 0x1F`
- `#define VCC4_SYSTEM_ERR (1<<8)`
- `#define VCC4_BUSY_ERR (1<<12)`
- `#define VCC4_COMMAND_ERR (1<<13)`
- `#define VCC4_PARAM_ERR (1<<14)`
- `#define VCC4_MODE_ERR (1<<15)`
- `#define VCC4_CAMERA_ERRS`
- `#define VCC4_NUM_OVERFLOW -1`
- `#define VCC4_BUF_UNDERFLOW -2`

### 11.521.1 Detailed Description

Interface with pan-tilt mechanism of a Canon VC-C4 camera

Definition in file [VCC4.H](#).

### 11.521.2 Define Documentation

#### 11.521.2.1 `#define VCC4_CAMERA_ERRS`

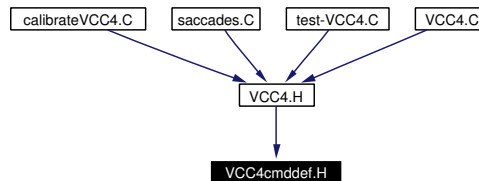
**Value:**

```
(VCC4_SYSTEM_ERR|VCC4_BUSY_ERR|VCC4_COMMAND_ERR|\
VCC4_PARAM_ERR|VCC4_MODE_ERR)
```

Definition at line 68 of file VCC4.H.

## 11.522 VCC4cmddef.H File Reference

This graph shows which files directly or indirectly include this file:



### Classes

- struct **VCC4CMD**

### Defines

- #define **VCC4\_SetCameraPowerOFF** 2
- #define **VCC4\_SetCameraPowerON** 3
- #define **VCC4\_SetFocusModeAF** 4
- #define **VCC4\_SetFocusModeMANUAL** 5
- #define **VCC4\_SetFocusModeNEAR** 6
- #define **VCC4\_SetFocusModeFAR** 7
- #define **VCC4\_SetFocusPosition** 8
- #define **VCC4\_GetFoucsPosition** 9
- #define **VCC4\_SetOnePushAF** 10
- #define **VCC4\_SetZoomingSTOP** 11
- #define **VCC4\_SetZoomingWIDE** 12
- #define **VCC4\_SetZoomingTELE** 13
- #define **VCC4\_SetZoomingHIWIDE** 14
- #define **VCC4\_SetZoomingHITELE** 15
- #define **VCC4\_SetZoomPosition2** 16
- #define **VCC4\_GetZoomPosition2** 17
- #define **VCC4\_SetZoomPosition4** 18
- #define **VCC4\_GetZoomPosition4** 19
- #define **VCC4\_SetZoomSpeed** 20
- #define **VCC4\_GetZoomSpeed** 21
- #define **VCC4\_SetBackLightOFF** 22
- #define **VCC4\_SetBackLightON** 23
- #define **VCC4\_SetExposedAUTO** 24
- #define **VCC4\_SetExposedMANUAL** 25

- #define VCC4\_SetShutterSpeedPRO 26
- #define VCC4\_SetShutterSpeed3 28
- #define VCC4\_SetShutterSpeed60 29
- #define VCC4\_SetShutterSpeed100 30
- #define VCC4\_SetAGCGain 31
- #define VCC4\_GetAGCGain 32
- #define VCC4\_SetIris 33
- #define VCC4\_GetIris 34
- #define VCC4\_SetAEVal 35
- #define VCC4\_GetAEVal 36
- #define VCC4\_SetGainSel 37
- #define VCC4\_GetGainSel 38
- #define VCC4\_SetWhiteBalance2 39
- #define VCC4\_SetWhiteBalance3 40
- #define VCC4\_SetWhiteBalance4 41
- #define VCC4\_GetWhiteBalance 42
- #define VCC4\_SetFading 44
- #define VCC4\_WhiteFade 45
- #define VCC4\_HiSpeedWhiteFade 46
- #define VCC4\_HiSpeedBlackFade 47
- #define VCC4\_CameraReset 48
- #define VCC4\_GetZoomRatio 49
- #define VCC4\_GetCCDSIZE 50
- #define VCC4\_GetZoomMax 53
- #define VCC4\_GetCameraVersion 57
- #define VCC4\_GetEEPROMVersion 58
- #define VCC4\_SetMotorSpeedPAN 59
- #define VCC4\_SetMotorSpeedTILT 60
- #define VCC4\_GetMotorSpeedPAN 61
- #define VCC4\_GetMotorSpeedTILT 62
- #define VCC4\_SetPanTilt1 63
- #define VCC4\_SetPanTilt2 64
- #define VCC4\_SetPanTilt3 65
- #define VCC4\_SetPanTilt4 66
- #define VCC4\_SetPanTilt5 67
- #define VCC4\_GoHome 69
- #define VCC4\_PTinit1 70
- #define VCC4\_PTinit2 71
- #define VCC4\_GetPanMoter0 72
- #define VCC4\_GetPanMoter1 73
- #define VCC4\_GetTiltMotor0 74
- #define VCC4\_GetTiltMotor1 75
- #define VCC4\_GetPanGRatio 76

- `#define VCC4_GetMotorSpeed` 77
- `#define VCC4_GetAngleRatio` 78
- `#define VCC4_GetAngleRatio2` 79
- `#define VCC4_GetAngleRatio0` 80
- `#define VCC4_GetAngleRatio1` 81
- `#define VCC4_PTStop` 82
- `#define VCC4_TiltUp` 83
- `#define VCC4_TiltDown` 84
- `#define VCC4_PanRigth` 85
- `#define VCC4_PanLeft` 86
- `#define VCC4_PanRightTiltUP` 87
- `#define VCC4_PanRightTiltDOWN` 88
- `#define VCC4_PanLeftTiltUP` 89
- `#define VCC4_PanLeftTiltDOWN` 90
- `#define VCC4_SetPTAngle2` 92
- `#define VCC4_GetPTAngle2` 93
- `#define VCC4_SetPTAngle0` 94
- `#define VCC4_SetPTAngle1` 95
- `#define VCC4_GetPTAngle0` 96
- `#define VCC4_GetPTAngle1` 97
- `#define VCC4_SerRCON` 102
- `#define VCC4_SetRCOFF` 103
- `#define VCC4_GetStatus` 104
- `#define VCC4_GetUnitName` 106
- `#define VCC4_GetRomVersion` 107
- `#define VCC4_SetPresetMemory` 108
- `#define VCC4_MovePreset` 109
- `#define VCC4_GetPresetStatus` 110
- `#define VCC4_RemoteCommandPass0` 113
- `#define VCC4_RemoteCommandPassN` 114
- `#define VCC4_SetCascadeON` 118
- `#define VCC4_SetCascadeOFF` 119
- `#define VCC4_EnterHostMode` 120
- `#define VCC4_EnterLocalMode` 121
- `#define VCC4_SetScreenOFF` 122
- `#define VCC4_SetScreenON` 123
- `#define VCC4_ScreenTitleOFF` 124
- `#define VCC4_ScreenTitleON` 125
- `#define VCC4_SetTimeMode0` 126
- `#define VCC4_SetTimeMode1` 127
- `#define VCC4_SetTimeMode2` 128
- `#define VCC4_SetDateMode0` 129
- `#define VCC4_SetDateMode1` 130

- #define **VCC4\_SetDateMode2** 131
- #define **VCC4\_SetTitleString** 132
- #define **VCC4\_GetTitleString** 133
- #define **VCC4\_SetDate** 134
- #define **VCC4\_GetDate** 135
- #define **VCC4\_SetTime** 136
- #define **VCC4\_GetTime** 137
- #define **VCC4\_GetCameraOnTime** 138
- #define **VCC4\_GetPedestalOnTime** 139
- #define **VCC4\_DefaultReset** 140
- #define **VCC4\_GetExtMove** 147
- #define **VCC4\_GetExtPreset** 148
- #define **VCC4\_SpecifyShutterSpeed** 149
- #define **VCC4\_SetWhiteBalance0** 150
- #define **VCC4\_SetWhiteBalance1** 151
- #define **VCC4\_GetFocusRange** 152
- #define **VCC4\_SetNotifyOFF** 155
- #define **VCC4\_SetNotifyON** 156
- #define **VCC4\_SetGlobalNotifyOFF** 157
- #define **VCC4\_SetGlobalNotifyON** 158
- #define **VCC4\_SetAELockOFF** 159
- #define **VCC4\_SetAELockON** 160
- #define **VCC4\_SetLedNormal** 164
- #define **VCC4\_SetLedGreenOn** 165
- #define **VCC4\_SetLedAllOff** 166
- #define **VCC4\_SetLedRedOn** 167
- #define **VCC4\_SetLedOrangeOn** 168
- #define **VCC4\_GetPedestalModel** 170
- #define **VCC4\_GetCameraModel** 171
- #define **VCC4\_CMDMAX** 139

### 11.522.1 Detailed Description

command definitions for the Canon VC-C4 camera

Definition in file [VCC4cmddef.H](#).

## 11.523 videodiff.C File Reference

```
#include "saliency.H"
#include "ModelManager.H"
#include "MPEGStream.H"
#include "ModelOptionDefs.H"
#include "Image_MathOps.H"
```

Include dependency graph for videodiff.C:

### Functions

- int **main** (const int argc, const char \*\*argv)

#### 11.523.1 Detailed Description

compute average difference between video frames

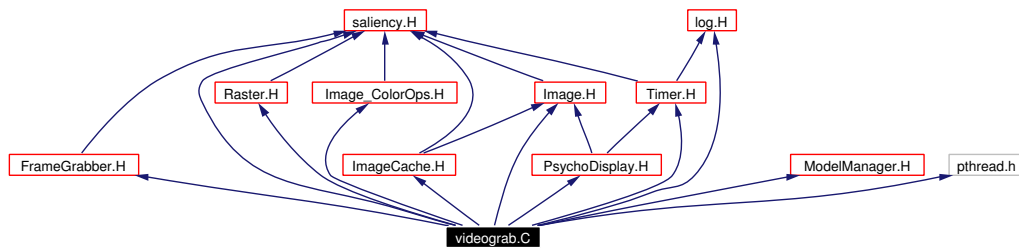
Definition in file [videodiff.C](#).



## 11.524 videograb.C File Reference

```
#include "saliency.H"
#include "Image.H"
#include "FrameGrabber.H"
#include "Timer.H"
#include "log.H"
#include "ModelManager.H"
#include "Raster.H"
#include "PsychoDisplay.H"
#include "ImageCache.H"
#include "Image_ColorOps.H"
#include <pthread.h>
```

Include dependency graph for videograb.C:



### Defines

- #define **NAVG** 20  
*number of frames over which frame rate is computed*
- #define **MAXQLEN** 1000  
*Maximum number of frames in queue.*

### Functions

- void \* **saveframes** (void \*)
- int **main** (const int argc, const char \*\*argv)

## Variables

- `pthread_mutex_t qmutex`
- `ImageCache< PixRGB< byte > > cache`
- `std::string base`

### 11.524.1 Detailed Description

grab frames and save them to disk

Definition in file [videograb.C](#).

### 11.524.2 Define Documentation

#### 11.524.2.1 `#define MAXQLEN 1000`

Maximum number of frames in queue.

Definition at line 53 of file [videograb.C](#).

#### 11.524.2.2 `#define NAVG 20`

number of frames over which frame rate is computed

Definition at line 50 of file [videograb.C](#).

### 11.524.3 Function Documentation

#### 11.524.3.1 `int main (const int argc, const char ** argv)`

This simple executable grabs video frames through the video4linux driver (see [V4Lgrabber.H](#)) or the IEEE1394 (firewire) grabber (see [IEEE1394grabber.H](#)). Selection of the grabber type is made via the `-fg-type=XX` command-line option. Frames are pushed into a queue and a second thread then tries to empty the queue as quickly as possible by writing the frames to disk. In testing, PPM format actually gave better frame rates than PNG, so that's what is used. Press to start grabbing and again to stop.

Definition at line 89 of file [videograb.C](#).

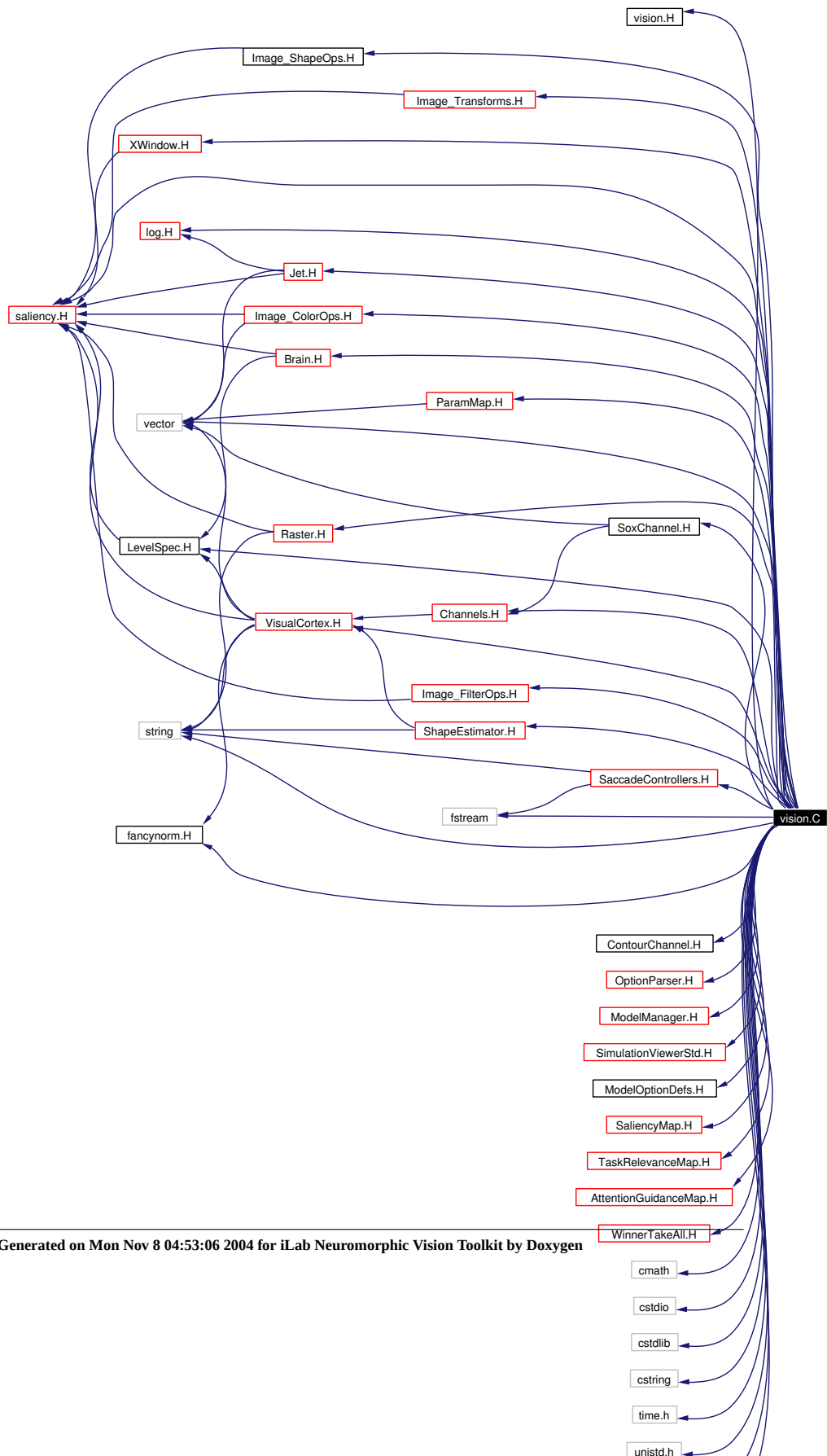
References `Timer::get()`, `SharedPtr< T >::get()`, `main()`, `NAVG`, `Timer::reset()`, `toYUV422()`, `uint64`, and `Image< PixRGB< byte > >::writeText()`.

## 11.525 vision.C File Reference

```
#include "vision.H"
#include "Brain.H"
#include "Channels.H"
#include "ContourChannel.H"
#include "fancynorm.H"
#include "Image_ColorOps.H"
#include "Image_FilterOps.H"
#include "Image_ShapeOps.H"
#include "Image_Transforms.H"
#include "LevelSpec.H"
#include "log.H"
#include "OptionParser.H"
#include "ParamMap.H"
#include "Raster.H"
#include "saliency.H"
#include "SoxChannel.H"
#include "VisualCortex.H"
#include "XWindow.H"
#include "Jet.H"
#include "SaccadeControllers.H"
#include "ModelManager.H"
#include "SimulationViewerStd.H"
#include "ShapeEstimator.H"
#include "ModelOptionDefs.H"
#include "SaliencyMap.H"
#include "TaskRelevanceMap.H"
#include "AttentionGuidanceMap.H"
#include "WinnerTakeAll.H"
#include <cmath>
```

```
#include <stdio>
#include <stdlib>
#include <string>
#include <time.h>
#include <unistd.h>
#include <signal.h>
#include <string>
#include <vector>
#include <fstream>
#include <iostream>
```

Include dependency graph for vision.C:



## Classes

- class [AppContext](#)  
*Manages state information for vision.C::main().*
- class [OldFrameSeries](#)  
*Manages a series of input or output frames for vision.C::main().*

## Defines

- #define **SMDnothing** 0x0000
- #define **SMDfoa** 0x0001
- #define **SMDpatch** 0x0002
- #define **SMDtraj** 0x0004
- #define **SMDtime** 0x0008
- #define **SMDadditive** 0x0010
- #define **SMDobject** 0x0020
- #define **SMDhilit** 0x0040
- #define **SMDshapeEstimator** 0x0080
- #define **SMDboring** 0x0100
- #define **SMDshapeEstimatorUseSM** 0x0200
- #define **SMDfoveateTraj** 0x0400
- #define **SMDnoIOR** 0x0800
- #define **SMDfoveateIn** 0x1000
- #define **SMDkillStatic** 0x2000
- #define **SMDall** 0xffff
- #define [TOO\\_MUCH\\_TIME](#) 500000  
*Quit if the salmap evolves more than TOO\_MUCH\_TIME ms.*
- #define [TOO\\_MANY\\_SHIFTS](#) 50000  
*Quit after TOO\_MANY\_SHIFTS of attention.*
- #define [XWW](#) 1024  
*Size of X window for interactive display:.*
- #define **XWH** 768
- #define [XWFAC](#) 100000.0  
*Saliency map factor for Xwindow display:.*
- #define [XWREFRESH](#) 30  
*Refresh rate (in ms) for [XWindow](#) display:.*

## Enumerations

- enum **State**

## Functions

- void \* **alloc\_mem** (size\_t bytes)
- void **dealloc\_mem** (void \*space)
- void \* **operator new** (size\_t bytes)
- void **operator delete** (void \*space)
- void \* **operator new[ ]** (size\_t bytes)
- void **operator delete[ ]** (void \*space)
- void **usage** (void)
- void **terminate** (int s)

*Signal handler (e.g., for control-C).*

- int **main** (int argc, const char \*\*argv)

*The main routine. Read images, process, display attended locations.*

## Variables

- int **MEM\_DEBUG\_CTR** = 0
- bool **keep\_going** = true

*Will turn false on interrupt signal.*

### 11.525.1 Detailed Description

This is the main bottomup attention executable. For command-line arguments, try running the program without any argument. A good starting point is to try "vision -T image.ppm xxx"

Definition in file [vision.C](#).

### 11.525.2 Define Documentation

#### 11.525.2.1 #define TOO\_MANY\_SHIFTS 50000

Quit after TOO\_MANY\_SHIFTS of attention.

Definition at line 148 of file vision.C.

Referenced by main().

**11.525.2.2 #define TOO\_MUCH\_TIME 500000**

Quit if the salmap evolves more than TOO\_MUCH\_TIME ms.

Definition at line 145 of file vision.C.

**11.525.2.3 #define XWFAC 100000.0**

Saliency map factor for Xwindow display:.

Definition at line 155 of file vision.C.

Referenced by main().

**11.525.2.4 #define XWREFRESH 30**

Refresh rate (in ms) for [XWindow](#) display:.

Definition at line 158 of file vision.C.

Referenced by main().

**11.525.2.5 #define XWW 1024**

Size of X window for interactive display:.

Definition at line 151 of file vision.C.

Referenced by SimulationViewerStd::drawMegaCombo(), and main().

**11.525.3 Function Documentation****11.525.3.1 int main (int *argc*, const char \*\* *argv*)**

The main routine. Read images, process, display attended locations.

Definition at line 427 of file vision.C.

References OptionSet::add(), OptionParser::addAlias(), GaborChannel::angle(), AS-SERT, Image< T >::clamp(), concatX(), concatY(), AppContext::contour\_channel, Image< T >::countParticles(), AppContext::crop\_dims, OldFrameSeries::delay(), OptionParser::destroy(), Image< T >::drawArrow(), Image< T >::drawCircle(), Image< T >::drawGrid(), XWindow::drawImage(), Image< T >::drawRect(), Image< T >::embed(), AppContext::embed, AppContext::exit\_after\_display, AppContext::foaCrop(), OldFrameSeries::frame(), FULLTRACE, SharedPtr< T >::get(), OptionParser::getCurrentOptArg(), Image< T >::getDims(), Image< T >::getHeight(), OptionParser::getNextOpt(), OptionParser::getOneLeftoverArg(), Image<



T >::getWidth(), Dims::h(), Point2D::i, AppContext::is\_interactive, AppContext::isCropping(), OldFrameSeries::isMultiframe(), Point2DT::isValid(), Point2D::j, keep\_going, lowPass9(), luminance(), main(), makeRGB(), MaxNormType, OldFrameSeries::minframe(), Rectangle::modify(), AppContext::nborient, NO\_INIT, Image< T >::normalize(), AppContext::normalizeSalmap(), Raster::NoXV(), OPTEXP\_ALL, Point2DT::p, OptionParser::printError(), OptionParser::printUsage(), AppContext::qlen, Raster::ReadGray(), Raster::ReadRGB(), rescale(), SharedPtr< T >::reset(), Image< T >::resize(), AppContext::resize\_dims, AppContext::resizeImage(), AppContext::salmap\_factor, AppContext::save\_output, AppContext::save\_submaps, AppContext::saveChannelOutput(), AppContext::saveCS(), AppContext::saveLandmark(), AppContext::saveShapeEstimator, scaleBlock(), OldFrameSeries::scan(), ModelManager::setOptionValString(), OptionParser::setOtherHelp(), Image< T >::setVal(), SM\_3D\_PITCH, AppContext::sox\_channel, terminate(), TOO\_MANY\_SHIFTS, uint, OldFrameSeries::update(), VCXNORM\_DEFAULT, VCXNORM\_FANCYONE, VCXNORM\_LANDMARK, VCXNORM\_MAXNORM, VCXNORM\_NONE, Raster::Visu(), Dims::w(), AppContext::waitKbd(), warp3D(), Raster::WriteFloat(), Raster::WriteGray(), Raster::WriteRGB(), Image< T >::writeText(), XWFAC, XWREFRESH, XWW, and ZEROS.

### 11.525.3.2 void terminate (int s)

Signal handler (e.g., for control-C).

Definition at line 166 of file vision.C.

References keep\_going.

## 11.525.4 Variable Documentation

### 11.525.4.1 bool keep\_going = true [static]

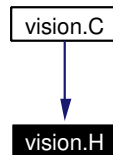
Will turn false on interrupt signal.

Definition at line 163 of file vision.C.

Referenced by main(), and terminate().

## 11.526 vision.H File Reference

This graph shows which files directly or indirectly include this file:



### Defines

- #define **SMOPT\_SHOWBY** 1
- #define **SMOPT\_SHOWRG** (1<<1)
- #define **SMOPT\_SHOWI** (1<<2)
- #define **SMOPT\_SHOWO** (1<<3)
- #define **SMOPT\_SHOWTRM** (1<<4)
- #define **SMOPT\_SHOWAGM** (1<<5)
- #define **SMOPT\_SHOWM** (1<<6)
- #define **SMOPT\_SHOWSM** (1<<7)
- #define **SMOPT\_SHOWLUM** (1<<9)
- #define **SMOPT\_SHOWTRAJ** (1<<10)
- #define **SMOPT\_SHOWCOMBX** (1<<11)
- #define **SMOPT\_SHOWCOMBY** (1<<12)
- #define **SMOPT\_FOREVER** (1<<13)
- #define **SMOPT\_JUSTSAVE** (1<<14)
- #define **SMOPT\_TARGET** (1<<15)
- #define **SMOPT\_NORM** (1<<20)
- #define **SMOPT\_3D** (1<<21)
- #define **SMOPT\_PEDES** (1<<24)
- #define **SMOPT\_MOTION** (1<<25)
- #define **SMOPT\_SHOWMT** (1<<26)
- #define **SMOPT\_LONGRANGE** (1<<27)
- #define **SMOPT\_SIXPACK** (1<<28)
- #define **SMOPT\_SHOWF** (1<<29)
- #define **SMOPT\_NERDCAM** (1<<30)
- #define **SMOPT\_SHOWLANDMARK** (1<<31)
- #define **SM\_3D\_PITCH** 25.0
- #define **SM\_3D\_YAW** 15.0

## 11.526.1 Detailed Description

Definitions of constants used in [vision.C](#)

Definition in file [vision.H](#).

## 11.526.2 Define Documentation

### 11.526.2.1 `#define SM_3D_PITCH 25.0`

angles for 3D display. Put POSITIVE values here, although we usually want negative values. cf [Image\\_ColorOps::warp3D](#) for the meaning of these angles

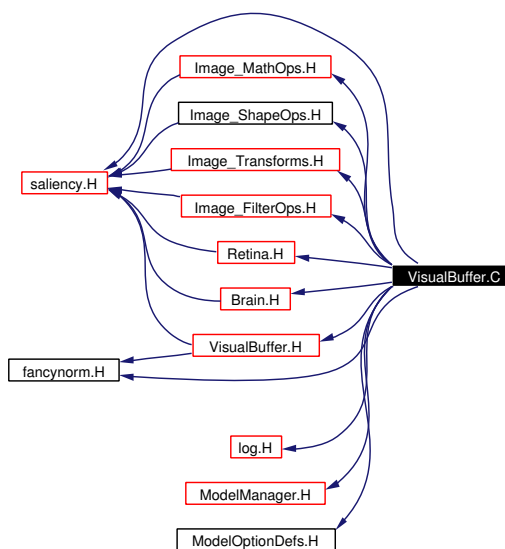
Definition at line 72 of file vision.H.

Referenced by `main()`.

## 11.527 VisualBuffer.C File Reference

```
#include "saliency.H"  
#include "log.H"  
#include "VisualBuffer.H"  
#include "ModelManager.H"  
#include "ModelOptionDefs.H"  
#include "Image_MathOps.H"  
#include "Image_ShapeOps.H"  
#include "Image_Transforms.H"  
#include "Image_FilterOps.H"  
#include "fancynorm.H"  
#include "Retina.H"  
#include "Brain.H"
```

Include dependency graph for VisualBuffer.C:



### 11.527.1 Detailed Description

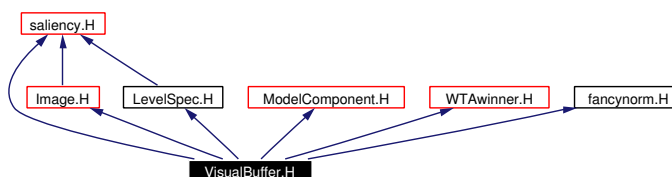
Grab ausio samples from /dev/dsp

Definition in file [VisualBuffer.C](#).

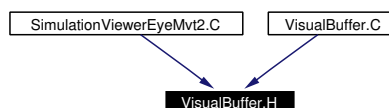
## 11.528 VisualBuffer.H File Reference

```
#include "saliency.H"
#include "ModelComponent.H"
#include "Image.H"
#include "WTAWinner.H"
#include "LevelSpec.H"
#include "fancynorm.H"
```

Include dependency graph for VisualBuffer.H:



This graph shows which files directly or indirectly include this file:



### Classes

- class [VisualBuffer](#)  
*An integrative visual buffer.*

### 11.528.1 Detailed Description

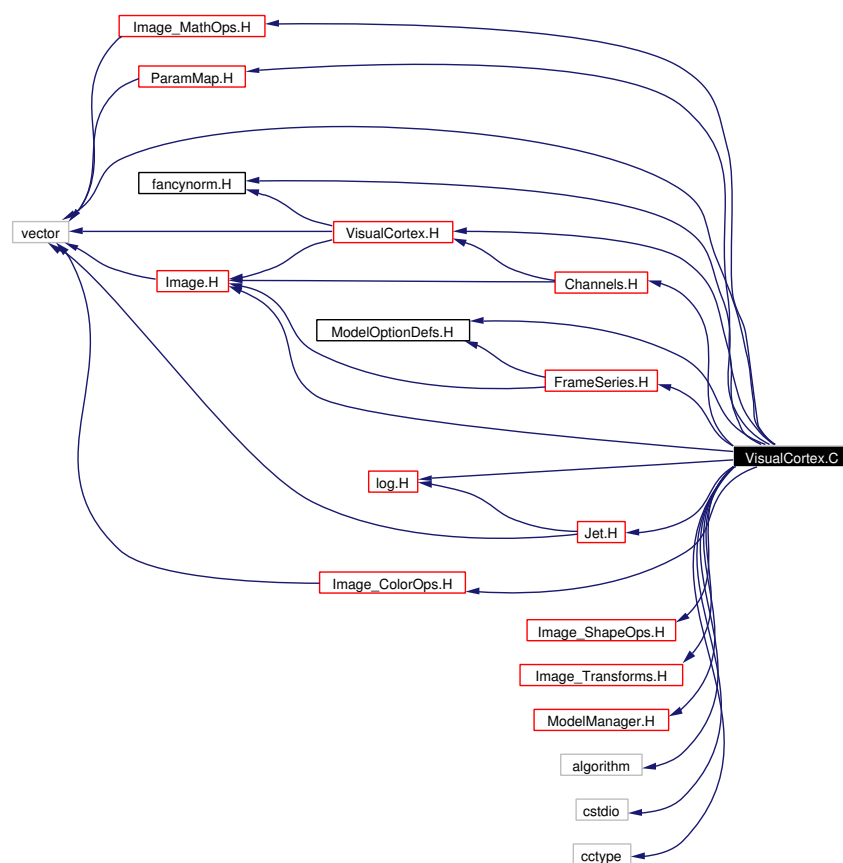
Grab audio samples from /dev/dsp

Definition in file [VisualBuffer.H](#).

## 11.529 VisualCortex.C File Reference

```
#include "VisualCortex.H"
#include "Channels.H"
#include "fancynorm.H"
#include "Image.H"
#include "Image_ColorOps.H"
#include "Image_MathOps.H"
#include "Image_ShapeOps.H"
#include "Image_Transforms.H"
#include "log.H"
#include "ParamMap.H"
#include "Jet.H"
#include "ModelManager.H"
#include "FrameSeries.H"
#include "ModelOptionDefs.H"
#include <algorithm>
#include <cstdio>
#include <cctype>
#include <vector>
```

Include dependency graph for VisualCortex.C:



### 11.529.1 Detailed Description

Implementation for visual cortex class

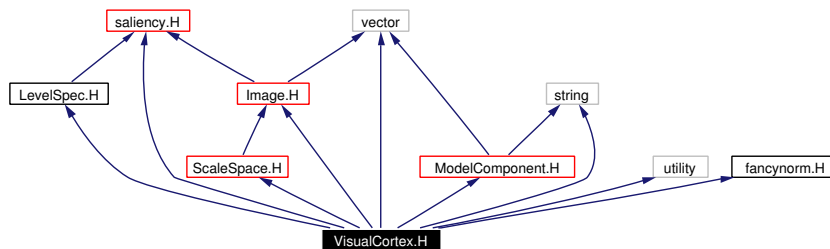
Definition in file [VisualCortex.C](#).



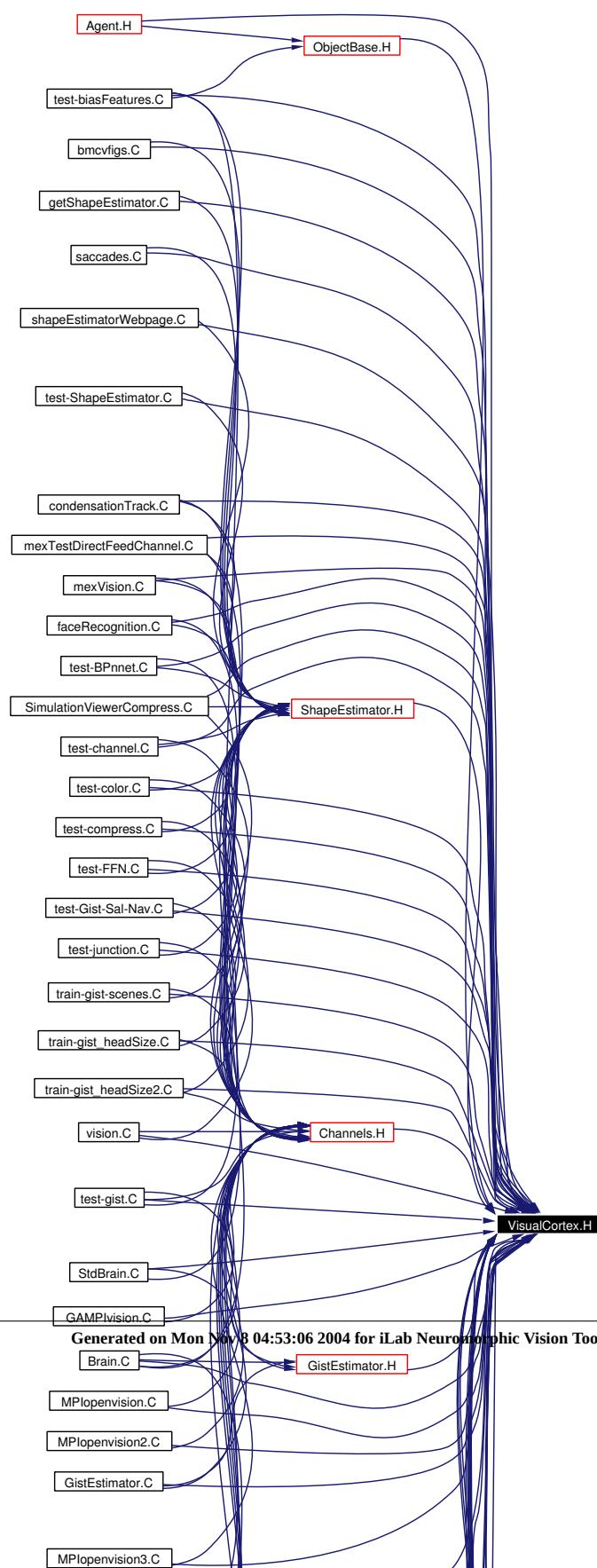
## 11.530 VisualCortex.H File Reference

```
#include "saliency.H"  
#include "Image.H"  
#include <string>  
#include <utility>  
#include <vector>  
#include "ModelComponent.H"  
#include "fancynorm.H"  
#include "LevelSpec.H"  
#include "ScaleSpace.H"
```

Include dependency graph for VisualCortex.H:



This graph shows which files directly or indirectly include this file:



## Classes

- class [VisualCortex](#)  
*The Visual Cortex Class.*

### 11.530.1 Detailed Description

The early visual processing stages

Definition in file [VisualCortex.H](#).

## 11.531 VisualCortexBeo.C File Reference

```
#include "VisualCortexBeo.H"
```

```
#include "ModelManager.H"
```

```
#include "Beowulf.H"
```

```
#include "ChannelsBeo.H"
```

Include dependency graph for VisualCortexBeo.C:

### 11.531.1 Detailed Description

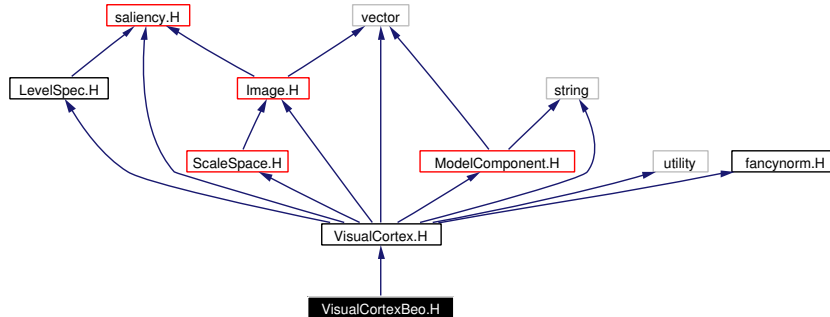
a [VisualCortex](#) with [SingleChannelBeo](#) channels

Definition in file [VisualCortexBeo.C](#).

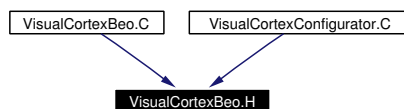
## 11.532 VisualCortexBeo.H File Reference

```
#include "VisualCortex.H"
```

Include dependency graph for VisualCortexBeo.H:



This graph shows which files directly or indirectly include this file:



### Classes

- class [VisualCortexBeo](#)

*This is a derivation of [VisualCortex](#) that uses [SingleChannelBeo](#) channels.*

### 11.532.1 Detailed Description

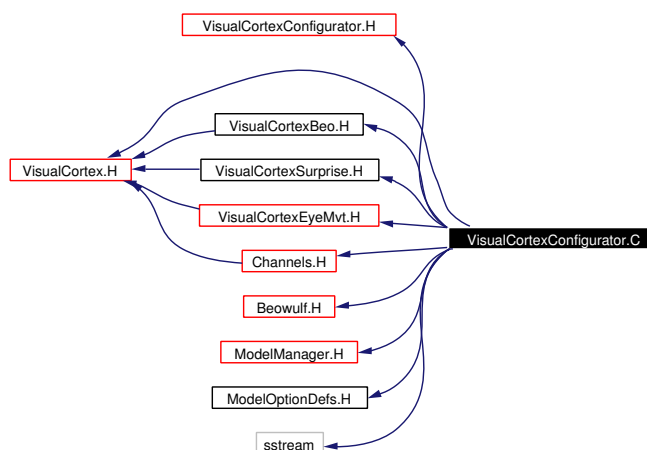
a [VisualCortex](#) with [SingleChannelBeo](#) channels

Definition in file [VisualCortexBeo.H](#).

## 11.533 VisualCortexConfigurator.C File Reference

```
#include "VisualCortexConfigurator.H"  
#include "VisualCortex.H"  
#include "VisualCortexBeo.H"  
#include "VisualCortexSurprise.H"  
#include "VisualCortexEyeMvt.H"  
#include "Beowulf.H"  
#include "ModelManager.H"  
#include "ModelOptionDefs.H"  
#include "Channels.H"  
#include <sstream>
```

Include dependency graph for VisualCortexConfigurator.C:



### 11.533.1 Detailed Description

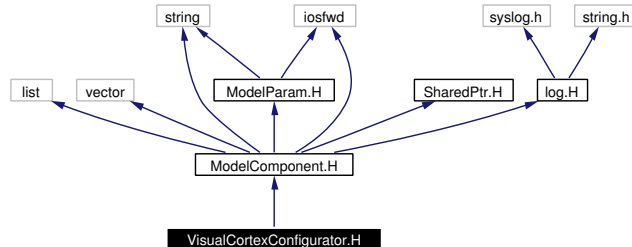
Class to select a [VisualCortex](#) at runtime

Definition in file [VisualCortexConfigurator.C](#).

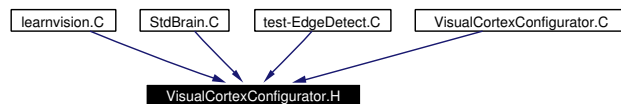
## 11.534 VisualCortexConfigurator.H File Reference

```
#include "ModelComponent.H"
```

Include dependency graph for VisualCortexConfigurator.H:



This graph shows which files directly or indirectly include this file:



### Classes

- class [VisualCortexConfigurator](#)  
*VisualCortex configurator.*

### 11.534.1 Detailed Description

run-time selects which VisualCortex to use

Definition in file [VisualCortexConfigurator.H](#).

## 11.535 VisualCortexEyeMvt.C File Reference

```
#include "VisualCortexEyeMvt.H"
#include "Channels.H"
#include "fancynorm.H"
#include "Image.H"
#include "Image_ColorOps.H"
#include "Image_MathOps.H"
#include "Image_ShapeOps.H"
#include "Image_FilterOps.H"
#include "Image_Transforms.H"
#include "log.H"
#include "ModelManager.H"
#include "ModelOptionDefs.H"
#include "Brain.H"
#include <algorithm>
#include <cstdio>
#include <cctype>
#include <vector>
```

Include dependency graph for VisualCortexEyeMvt.C:

### 11.535.1 Detailed Description

Implementation for the fake human-eye-movement visual cortex class

Definition in file [VisualCortexEyeMvt.C](#).

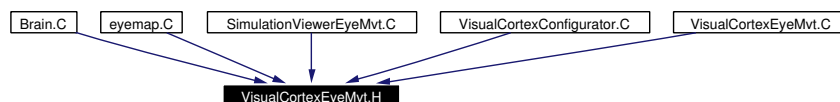


## 11.536 VisualCortexEyeMvt.H File Reference

```
#include "VisualCortex.H"  
#include "ImageSet.H"  
#include <vector>  
#include "SaccadeControllers.H"
```

Include dependency graph for VisualCortexEyeMvt.H:

This graph shows which files directly or indirectly include this file:



### Classes

- class [VisualCortexEyeMvt](#)  
*The VisualCortexEyeMvt Class.*

### 11.536.1 Detailed Description

a fake [VisualCortex](#) whose saliency map is built from human eye movement traces

Definition in file [VisualCortexEyeMvt.H](#).

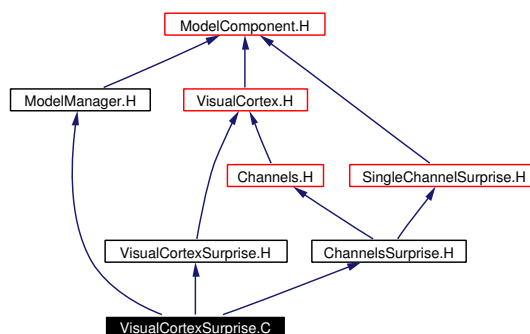
## 11.537 VisualCortexSurprise.C File Reference

```
#include "VisualCortexSurprise.H"
```

```
#include "ModelManager.H"
```

```
#include "ChannelsSurprise.H"
```

Include dependency graph for VisualCortexSurprise.C:



### 11.537.1 Detailed Description

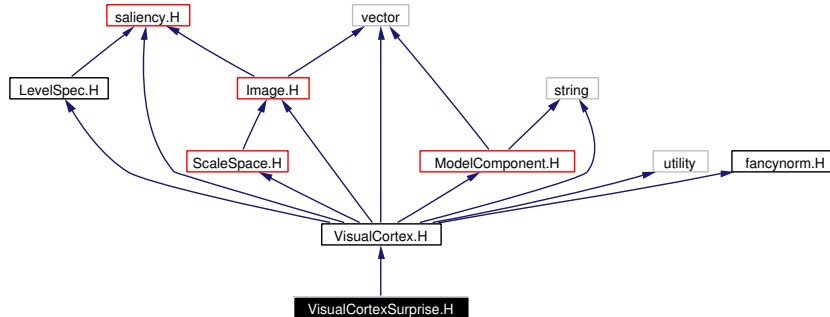
a [VisualCortex](#) with [SingleChannelSurprise](#) channels

Definition in file [VisualCortexSurprise.C](#).

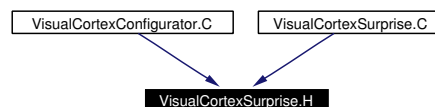
## 11.538 VisualCortexSurprise.H File Reference

```
#include "VisualCortex.H"
```

Include dependency graph for VisualCortexSurprise.H:



This graph shows which files directly or indirectly include this file:



### Classes

- class [VisualCortexSurprise](#)  
This is a derivation of [VisualCortex](#) using [SingleChannelSurprise](#) channels.

### 11.538.1 Detailed Description

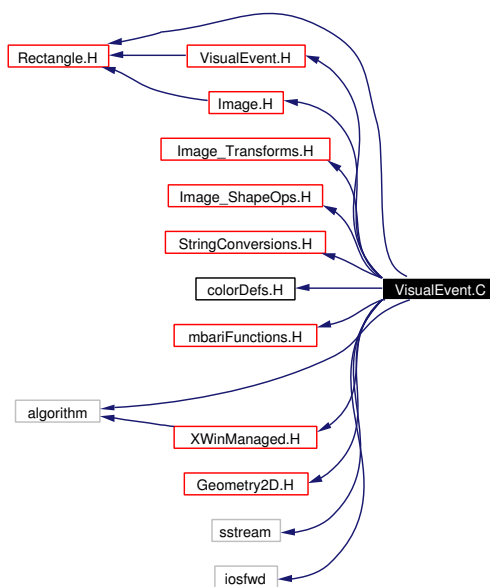
a [VisualCortex](#) with [SingleChannelSurprise](#) channels

Definition in file [VisualCortexSurprise.H](#).

## 11.539 VisualEvent.C File Reference

```
#include "VisualEvent.H"  
#include "Image.H"  
#include "Image_Transforms.H"  
#include "Image_ShapeOps.H"  
#include "StringConversions.H"  
#include "colorDefs.H"  
#include "Rectangle.H"  
#include "mbariFunctions.H"  
#include "XWinManaged.H"  
#include "Geometry2D.H"  
#include <sstream>  
#include <iosfwd>  
#include <algorithm>
```

Include dependency graph for VisualEvent.C:



### 11.539.1 Detailed Description

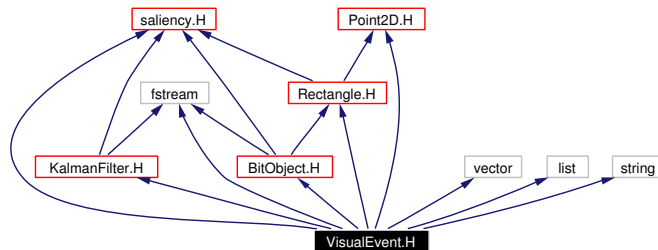
classes useful for event tracking

Definition in file [VisualEvent.C](#).

## 11.540 VisualEvent.H File Reference

```
#include "saliency.H"
#include "Point2D.H"
#include "Rectangle.H"
#include "BitObject.H"
#include "KalmanFilter.H"
#include <vector>
#include <list>
#include <string>
#include <fstream>
```

Include dependency graph for VisualEvent.H:



This graph shows which files directly or indirectly include this file:

### Classes

- class [Token](#)  
*public class that contains information for a visual token for tracking*
- class [PropertyVectorSet](#)  
*class that contains a set of property vectors for events*
- class [VisualEvent](#)  
*stores a visual event as a sequence of tokens with additional information*
- class [VisualEventSet](#)  
*contains a set of visual events to which a set of tokens can be matched*

### 11.540.1 Detailed Description

classes useful for event tracking

Definition in file [VisualEvent.H](#).

## 11.541 VisualFeatures.H File Reference

This graph shows which files directly or indirectly include this file:

### Defines

- #define **NBVISUALFEATURES** 39  
*Number of visual features.*
- #define **NBVISUALFEATURETYPES** 3
- #define **NBVISUALFEATURELEVELS** 3

### Enumerations

- enum **VisualFeature** {  
**UNKNOWN** = 0, **COMPOSITECOLOR** = 1, **COLOR** = 2, **SOCOLOR** = 3,  
**RG** = 4, **BY** = 5, **SORG** = 6, **SOGR** = 7,  
**SOBY** = 8, **SOYB** = 9, **R** = 10, **G** = 11,  
**B** = 12, **Y** = 13, **INTENS** = 14, **FLICKER** = 15,  
**ORI** = 16, **TEMPLATE** = 17, **COLBAND** = 18, **MOTION** = 19,  
**CONVOLVE** = 20, **LJUNCTION** = 21, **TJUNCTION** = 22, **XJUNCTION** = 23,  
**ENDPOINT** = 24, **RGBCONVOLVE** = 25, **ENTROPY** = 26, **DUMMY** = 27,  
**PN03CONTRAST** = 28, **DISPARITY** = 29, **STEREO** = 30, **INFORMATION** =  
31,  
**RECOGNITION** = 32, **HUE** = 33, **DIRECTFEED** = 34, **MULTIDIRECTFEED**  
= 35,  
**VARIANCE** = 36, **TCORR** = 37, **SCORR** = 38 }  
*Various features that can be represented.*
- enum **VisualFeatureType** { **RAW** = 0, **RAW\_CS** = 1, **NORM\_CS** = 2 }  
*Various types of feature representations.*
- enum **VisualFeatureHierarchy** { **V1** = 1, **V2** = 2, **V4** = 4 }

### Functions

- const char \* **featureName** (const **VisualFeature** f)  
*Access feature name in clear.*



- const char \* [featureTypeName](#) (const [VisualFeatureType](#) t)  
*Access feature type name in clear.*
- const int [featureHierarchyLevel](#) (const [VisualFeature](#) f)  
*Access feature level in the visual hierarchy.*

### 11.541.1 Detailed Description

definitions of known visual features

Definition in file [VisualFeatures.H](#).

### 11.541.2 Define Documentation

#### 11.541.2.1 `#define NBVISUALFEATURES 39`

Number of visual features.

Definition at line 89 of file [VisualFeatures.H](#).

Referenced by [convertFromString\(\)](#), [featureHierarchyLevel\(\)](#), [featureName\(\)](#), [JetSpec::freeMem\(\)](#), and [raodistance\(\)](#).

### 11.541.3 Enumeration Type Documentation

#### 11.541.3.1 enum [VisualFeature](#)

Various features that can be represented.

We assign a specific number to each definition to speedup name printing

#### Enumeration values:

**UNKNOWN** Unknown feature type.

**COMPOSITECOLOR** composite Color channel

**COLOR** double opponent Color complex feature

**SOCOLOR** single opponent Color complex feature

**RG** Red/green double opponency.

**BY** Blue/yellow double-opponency.

**SORG** Red/green single opponency.

**SOGR** Green/Red single opponency.

**SOBY** Blue/Yellow single opponency.

**SOYB** Yellow/Blue single opponency.

**R** Red channel.

**G** green channel

**B** Blue channel.

**Y** yellow channel

**INTENS** Intensity center-surround.

**FLICKER** Flicker center-surround.

**ORI** Orientation contrast.

**TEMPLATE** Template matching.

**COLBAND** Color band.

**MOTION** Motion channel.

**CONVOLVE** Convolution channel.

**LJUNCTION** L junction channel.

**TJUNCTION** T junction channel.

**XJUNCTION** X junction channel.

**ENDPOINT** End [Point](#) channel.

**RGBCONVOLVE** Color convolution channel.

**ENTROPY** Local entropy channel.

**DUMMY** Dummy channel.

**PN03CONTRAST** Local contrast (Parkhurst&Niebur 2003).

**DISPARITY** Disparity Channel.

**STEREO** Stereo channel.

**INFORMATION** Information channel.

**RECOGNITION** Recognition channel.

**HUE** Hue channel.

**DIRECTFEED** DirectFeed channel.

**MULTIDIRECTFEED** MultiDirectFeed channel.

**VARIANCE** Variance channel.

**TCORR** Tcorr (temporal correlation) channel.

**SCORR** Scorr (spatial correlation) channel.

Definition at line 42 of file VisualFeatures.H.

Referenced by ChannelBase::ChannelBase(), SingleChannelBeoServer::check(), convertFromString(), BeobotVisualCortex::masterCollect(), raodistance(), and ChannelBase::visualFeature().

### 11.541.3.2 enum [VisualFeatureHierarchy](#)

**Enumeration values:**

**V1** V1 channel in visual hierarchy.

**V2** V2 channel in visual hierarchy.

**V4** V4 channel in visual hierarchy.

Definition at line 103 of file VisualFeatures.H.

### 11.541.3.3 enum [VisualFeatureType](#)

Various types of feature representations.

We assign a specific number to each definition to speedup name printing

**Enumeration values:**

**RAW** Raw pixel data.

**RAW\_CS** Center-surround data before maxnormalization.

**NORM\_CS** Center-surround data after maxnormalization.

Definition at line 93 of file VisualFeatures.H.

## 11.541.4 Function Documentation

### 11.541.4.1 `const int featureHierarchyLevel (const VisualFeature f)` [inline]

Access feature level in the visual hierarchy.

Definition at line 151 of file VisualFeatures.H.

References NBVISUALFEATURES.

Referenced by VisualCortex::input().

### 11.541.4.2 `const char * featureName (const VisualFeature f)` [inline]

Access feature name in clear.

Definition at line 125 of file VisualFeatures.H.

References NBVISUALFEATURES.

Referenced by JetSpec::addFeature(), JetSpec::addFeatureType(), convertFromString(), convertToString(), JetSpec::getIndexA(), JetSpec::getIndexV(), and JetSpec::print().

**11.541.4.3** `const char * featureTypeName (const VisualFeatureType t)`  
[inline]

Access feature type name in clear.

Definition at line 143 of file VisualFeatures.H.

Referenced by `JetSpec::addFeatureType()`, `JetSpec::getIndexA()`, `JetSpec::getIndexV()`, and `JetSpec::print()`.

## 11.542 VisualObject.C File Reference

```
#include "VisualObject.H"  
#include "Histogram.H"  
#include "KDTree.H"  
#include "Image_ColorOps.H"  
#include "Image_ShapeOps.H"  
#include "Raster.H"  
#include "log.H"  
#include <cstring>  
#include <math.h>
```

Include dependency graph for VisualObject.C:

### 11.542.1 Detailed Description

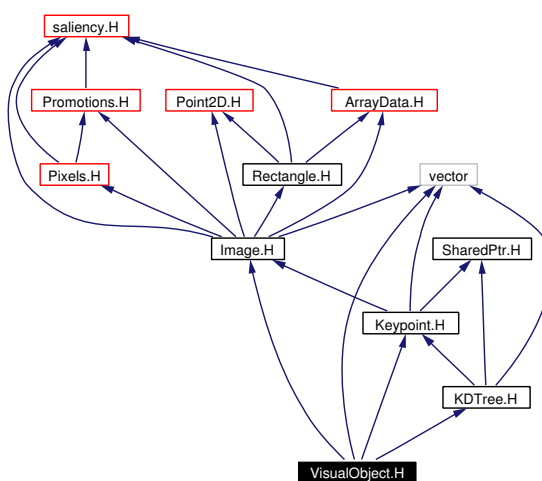
Visual Objects to be recognized

Definition in file [VisualObject.C](#).

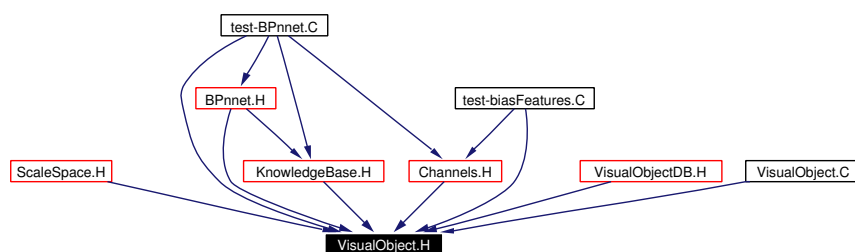
## 11.543 VisualObject.H File Reference

```
#include "Image.H"
#include "KDTree.H"
#include "Keypoint.H"
#include <vector>
```

Include dependency graph for VisualObject.H:



This graph shows which files directly or indirectly include this file:



## Classes

- class [KeypointMatch](#)  
*KeypointMatch.*

- class [VisualObjectMatch](#)  
*VisualObjectMatch.*
- class [VisualObject](#)  
*describes name and attributes of a visual object*

## Variables

- const int **ORIENTARRAY** = 36

### 11.543.1 Detailed Description

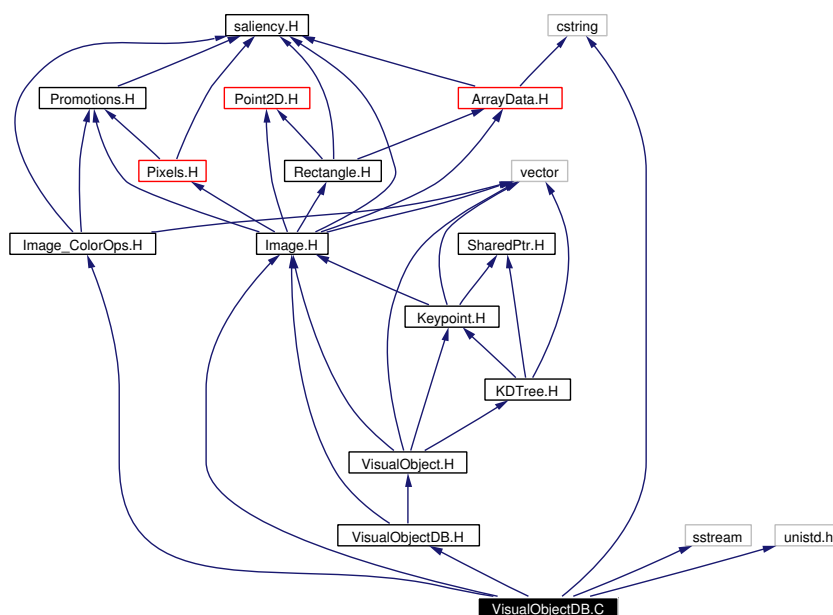
Header file for visual objects to be recognized

Definition in file [VisualObject.H](#).

## 11.544 VisualObjectDB.C File Reference

```
#include "VisualObjectDB.H"
#include "Image.H"
#include "Image_ColorOps.H"
#include <cstring>
#include <sstream>
#include <unistd.h>
```

Include dependency graph for VisualObjectDB.C:



### 11.544.1 Detailed Description

Visual object database

Definition in file [VisualObjectDB.C](#).

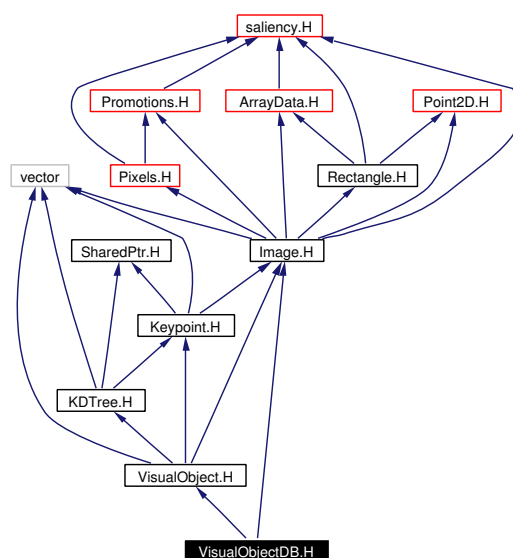


## 11.545 VisualObjectDB.H File Reference

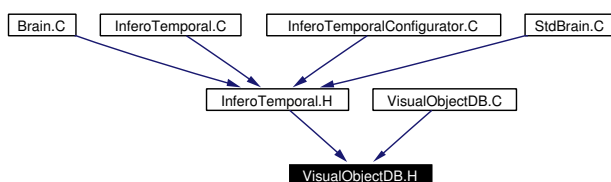
```
#include "VisualObject.H"
```

```
#include "Image.H"
```

Include dependency graph for VisualObjectDB.H:



This graph shows which files directly or indirectly include this file:



## Classes

- class `VisualObjectDB`  
`VisualObjectDatabase.`

### 11.545.1 Detailed Description

Visual object database

Definition in file [VisualObjectDB.H](#).

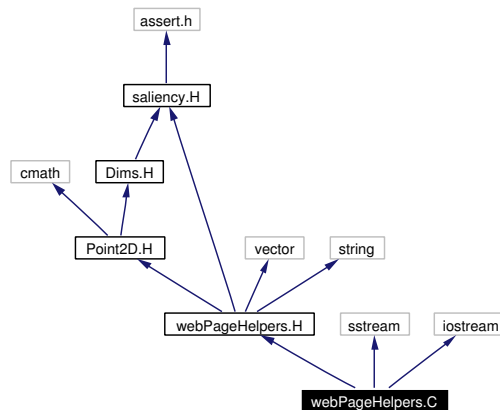
## 11.546 webPageHelpers.C File Reference

```
#include "webPageHelpers.H"
```

```
#include <sstream>
```

```
#include <iostream>
```

Include dependency graph for webPageHelpers.C:



## Functions

- [string cat](#) (const char \*s1, const char \*s2, const char \*s3, const char \*s4)  
*concatenate up to four char\* strings*
- const char \* [webHeader](#) (const char \*title)  
*return a string with the header info for a web page with title 'title'*
- const char \* [webFooter](#) ()  
*return a string with the footer info for a web page*
- const char \* [webChannellLegend](#) ()  
*return a string defining the legend used for the shapeEstimator web pages*
- const char \* [webImageStart](#) ()  
*return a string to start a two column table for images*
- const char \* [webImageEnd](#) ()  
*return a string to close a table for images*

- `const char * webImageTag (const char *filename, const char *caption, const char *link, int num)`  
*return a string with an image tag with caption*
- `const char * webHypLinkStart (const char *href)`  
*return a string to start a hyperlink with target 'href'*
- `const char * webHypLinkEnd ()`  
*return a string to close a hyperlink*
- `const char * webDetailName (const char *basename, const int num)`  
*append num as a string to basename*
- `const char * webDetailHeader (const char *basename, const int num)`  
*return a string with the header of a "details" page*
- `const char * webTime2str (const float stime)`  
*return a time given in s as a string in ms*

### 11.546.1 Detailed Description

helper functions for the creation of web pages

Definition in file [webPageHelpers.C](#).

### 11.546.2 Function Documentation

#### 11.546.2.1 `string cat (const char * s1, const char * s2, const char * s3, const char * s4)`

concatenate up to four char\* strings

Definition at line 44 of file [webPageHelpers.C](#).

#### 11.546.2.2 `const char* webChannelLegend ()`

return a string defining the legend used for the shapeEstimator web pages

Definition at line 72 of file [webPageHelpers.C](#).

**11.546.2.3   const char\* webDetailHeader (const char \* *basename*, const int *num*)**

return a string with the header of a "details" page

Definition at line 169 of file webPageHelpers.C.

References webHeader().

**11.546.2.4   const char\* webDetailName (const char \* *basename*, const int *num*)**

append num as a string to basename

Definition at line 161 of file webPageHelpers.C.

**11.546.2.5   const char\* webFooter ()**

return a string with the footer info for a web page

Definition at line 63 of file webPageHelpers.C.

**11.546.2.6   const char\* webHeader (const char \* *title*)**

return a string with the header info for a web page with title 'title'

Definition at line 52 of file webPageHelpers.C.

Referenced by webDetailHeader().

**11.546.2.7   const char\* webHypLinkEnd ()**

return a string to close a hyperlink

Definition at line 155 of file webPageHelpers.C.

Referenced by webImageTag().

**11.546.2.8   const char\* webHypLinkStart (const char \* *href*)**

return a string to start a hyperlink with target 'href'

Definition at line 147 of file webPageHelpers.C.

Referenced by webImageTag().

**11.546.2.9   const char\* webImageEnd ()**

return a string to close a table for images

Definition at line 95 of file webPageHelpers.C.

#### 11.546.2.10 **const char\* webImageStart ()**

return a string to start a two column table for images

Definition at line 85 of file webPageHelpers.C.

#### 11.546.2.11 **const char\* webImageTag (const char \* *filename*, const char \* *caption*, const char \* *link* = "", int *num* = -1)**

return a string with an image tag with caption

##### **Parameters:**

***filename*** filename of the image for the tag

***caption*** a string with the caption for the image

***link*** a target address to use the image as a link (default is "" = no link)

- num***
- (num < 0) - image is not part of an image table
  - (num >= 0) - image is part of an image table:
    - num odd - image is in the left column of the table
    - num even - image is in the right column of the table

Definition at line 101 of file webPageHelpers.C.

References webHypLinkEnd(), and webHypLinkStart().

#### 11.546.2.12 **const char\* webTime2str (const float *stime*)**

return a time given in s as a string in ms

Definition at line 177 of file webPageHelpers.C.

## 11.547 webPageHelpers.H File Reference

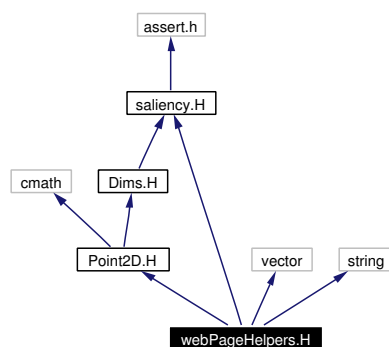
```
#include "saliency.H"
```

```
#include "Point2D.H"
```

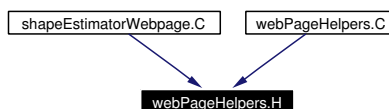
```
#include <vector>
```

```
#include <string>
```

Include dependency graph for webPageHelpers.H:



This graph shows which files directly or indirectly include this file:



## Functions

- `std::string cat` (const char \*s1="", const char \*s2="", const char \*s3="", const char \*s4="")  
*concatenate up to four char\* strings*
- const char \* `webHeader` (const char \*title)  
*return a string with the header info for a web page with title 'title'*
- const char \* `webFooter` ()  
*return a string with the footer info for a web page*

- const char \* [webChannelLegend](#) ()  
*return a string defining the legend used for the shapeEstimator web pages*
- const char \* [webImageStart](#) ()  
*return a string to start a two column table for images*
- const char \* [webImageEnd](#) ()  
*return a string to close a table for images*
- const char \* [webImageTag](#) (const char \*filename, const char \*caption, const char \*link="", int num=-1)  
*return a string with an image tag with caption*
- const char \* [webHypLinkStart](#) (const char \*href)  
*return a string to start a hyperlink with target 'href'*
- const char \* [webHypLinkEnd](#) ()  
*return a string to close a hyperlink*
- const char \* [webDetailName](#) (const char \*basename, const int num)  
*append num as a string to basename*
- const char \* [webDetailHeader](#) (const char \*basename, const int num)  
*return a string with the header of a "details" page*
- const char \* [webTime2str](#) (const float stime)  
*return a time given in s as a string in ms*

### 11.547.1 Detailed Description

helper functions for the creation of the shapeEstimator web pages

Definition in file [webPageHelpers.H](#).

### 11.547.2 Function Documentation

#### 11.547.2.1 [std::string](#) cat (const char \* s1 = "", const char \* s2 = "", const char \* s3 = "", const char \* s4 = "")

concatenate up to four char\* strings

Definition at line 44 of file [webPageHelpers.C](#).



**11.547.2.2 const char\* webChannelLegend ()**

return a string defining the legend used for the shapeEstimator web pages

Definition at line 72 of file webPageHelpers.C.

**11.547.2.3 const char\* webDetailHeader (const char \* *basename*, const int *num*)**

return a string with the header of a "details" page

Definition at line 169 of file webPageHelpers.C.

References webHeader().

**11.547.2.4 const char\* webDetailName (const char \* *basename*, const int *num*)**

append num as a string to basename

Definition at line 161 of file webPageHelpers.C.

**11.547.2.5 const char\* webFooter ()**

return a string with the footer info for a web page

Definition at line 63 of file webPageHelpers.C.

**11.547.2.6 const char\* webHeader (const char \* *title*)**

return a string with the header info for a web page with title 'title'

Definition at line 52 of file webPageHelpers.C.

Referenced by webDetailHeader().

**11.547.2.7 const char\* webHypLinkEnd ()**

return a string to close a hyperlink

Definition at line 155 of file webPageHelpers.C.

Referenced by webImageTag().

**11.547.2.8 const char\* webHypLinkStart (const char \* *href*)**

return a string to start a hyperlink with target 'href'

Definition at line 147 of file webPageHelpers.C.

Referenced by webImageTag().

#### 11.547.2.9 **const char\* webImageEnd ()**

return a string to close a table for images

Definition at line 95 of file webPageHelpers.C.

#### 11.547.2.10 **const char\* webImageStart ()**

return a string to start a two column table for images

Definition at line 85 of file webPageHelpers.C.

#### 11.547.2.11 **const char\* webImageTag (const char \* *filename*, const char \* *caption*, const char \* *link* = "", int *num* = -1)**

return a string with an image tag with caption

##### **Parameters:**

***filename*** filename of the image for the tag

***caption*** a string with the caption for the image

***link*** a target address to use the image as a link (default is "" = no link)

- num***
- (num < 0) - image is not part of an image table
  - (num >= 0) - image is part of an image table:
    - num odd - image is in the left column of the table
    - num even - image is in the right column of the table

Definition at line 101 of file webPageHelpers.C.

References webHypLinkEnd(), and webHypLinkStart().

#### 11.547.2.12 **const char\* webTime2str (const float *stime*)**

return a time given in s as a string in ms

Definition at line 177 of file webPageHelpers.C.

## 11.548 WinnerTakeAll.C File Reference

```
#include "WinnerTakeAll.H"  
#include "Image.H"  
#include "log.H"  
#include "ModelOptionDefs.H"  
#include "ModelManager.H"  
#include "FrameSeries.H"
```

Include dependency graph for WinnerTakeAll.C:

### 11.548.1 Detailed Description

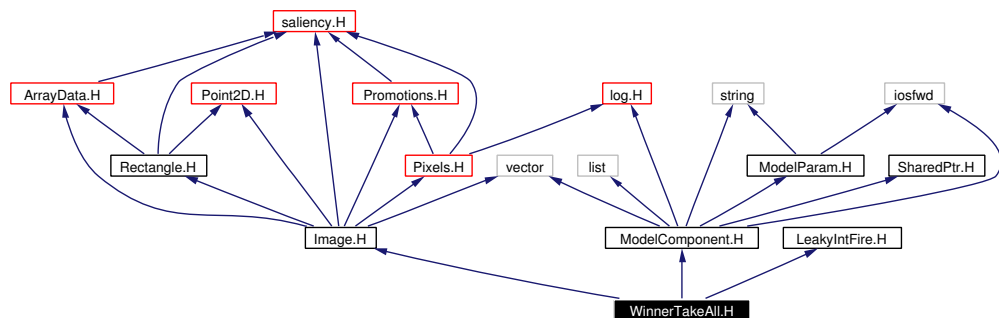
2D winner-take-all network

Definition in file [WinnerTakeAll.C](#).

## 11.549 WinnerTakeAll.H File Reference

```
#include "Image.H"
#include "LeakyIntFire.H"
#include "ModelComponent.H"
```

Include dependency graph for WinnerTakeAll.H:



This graph shows which files directly or indirectly include this file:

### Classes

- class [WinnerTakeAll](#)  
*The winner-take-all base class.*
- class [WinnerTakeAllConfigurator](#)  
*WinnerTakeAll configurator.*
- class [WinnerTakeAllStd](#)  
*The standard winner-take-all.*
- class [WinnerTakeAllFast](#)  
*A fast winner-take-all.*
- class [WinnerTakeAllDummy](#)  
*A dummy winner-take-all.*

### 11.549.1 Detailed Description

Header file for winner-take-all network

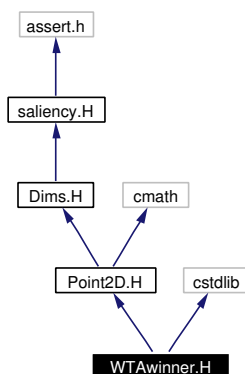
Definition in file [WinnerTakeAll.H](#).

## 11.550 WTAwinner.H File Reference

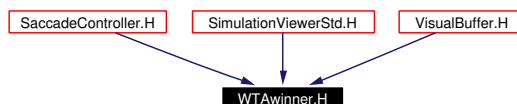
```
#include "Point2D.H"
```

```
#include <cstdlib>
```

Include dependency graph for WTAwinner.H:



This graph shows which files directly or indirectly include this file:



### Classes

- class [WTAwinner](#)

*This is an open class representing a WTA winner (covert attention shift).*

### 11.550.1 Detailed Description

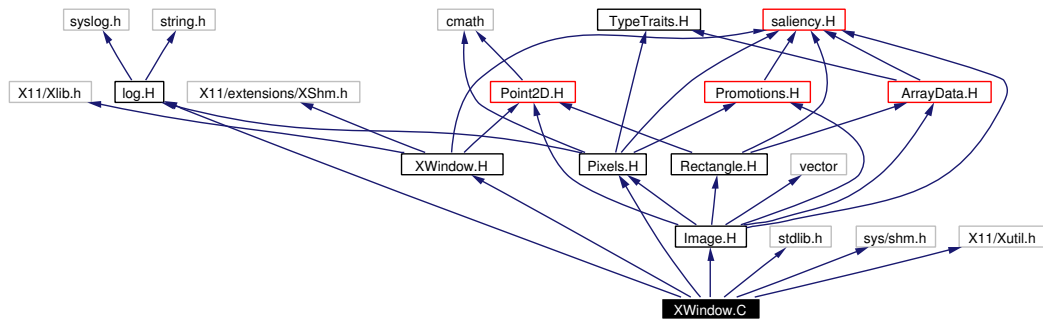
a winner-take-all winner (i.e., covert shift of attention)

Definition in file [WTAwinner.H](#).

## 11.551 XWindow.C File Reference

```
#include "XWindow.H"
#include "log.H"
#include "Pixels.H"
#include "Image.H"
#include <stdlib.h>
#include <sys/shm.h>
#include <X11/Xutil.h>
```

Include dependency graph for XWindow.C:



## Functions

- int **fourByteAlign** (int v)

### 11.551.1 Detailed Description

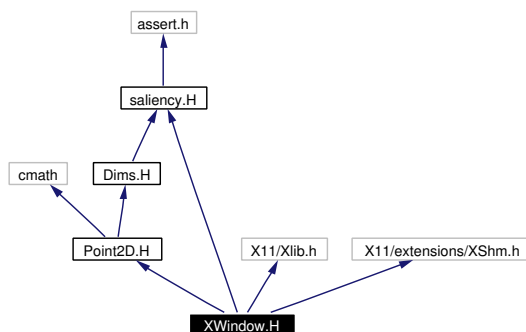
Implementation for a simple window class

Definition in file [XWindow.C](#).

## 11.552 XWindow.H File Reference

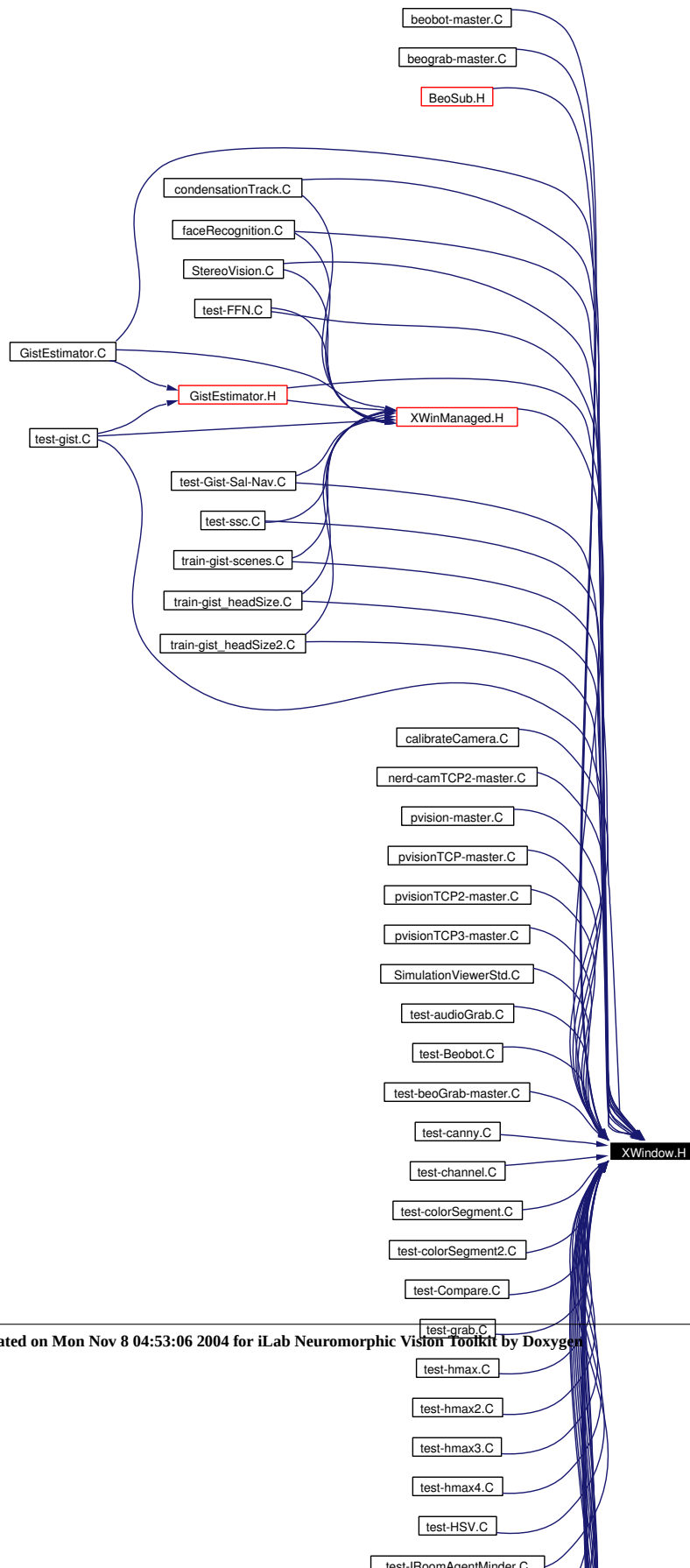
```
#include "saliency.H"  
#include "Point2D.H"  
#include <X11/Xlib.h>  
#include <X11/extensions/XShm.h>
```

Include dependency graph for XWindow.H:



This graph shows which files directly or indirectly include this file:





## Classes

- class [XWindow](#)

*A simple X-window to display images.*

### 11.552.1 Detailed Description

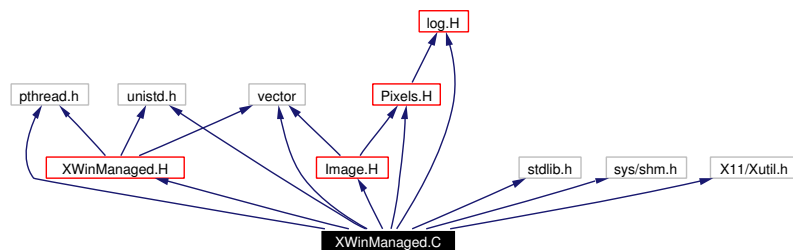
Class definition for a simple window

Definition in file [XWindow.H](#).

## 11.553 XWinManaged.C File Reference

```
#include "XWinManaged.H"  
#include "log.H"  
#include "Pixels.H"  
#include "Image.H"  
#include <unistd.h>  
#include <pthread.h>  
#include <stdlib.h>  
#include <sys/shm.h>  
#include <X11/Xutil.h>  
#include <vector>
```

Include dependency graph for XWinManaged.C:



### Defines

- #define SLEEPTIME 100000

### 11.553.1 Detailed Description

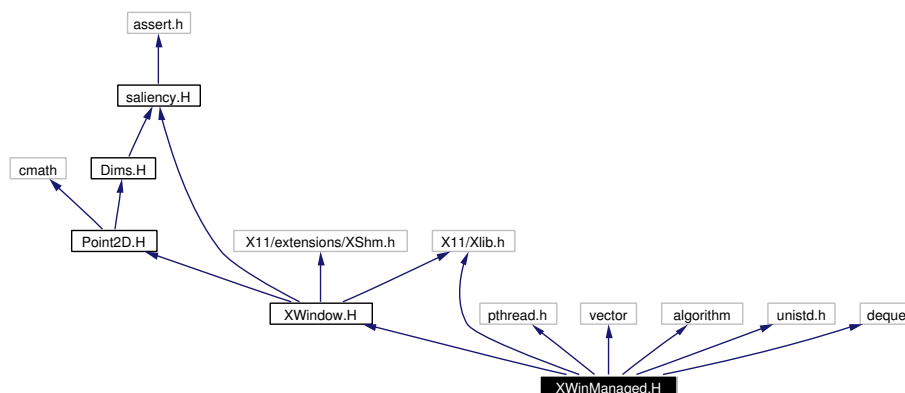
A window class with active window management

Definition in file [XWinManaged.C](#).

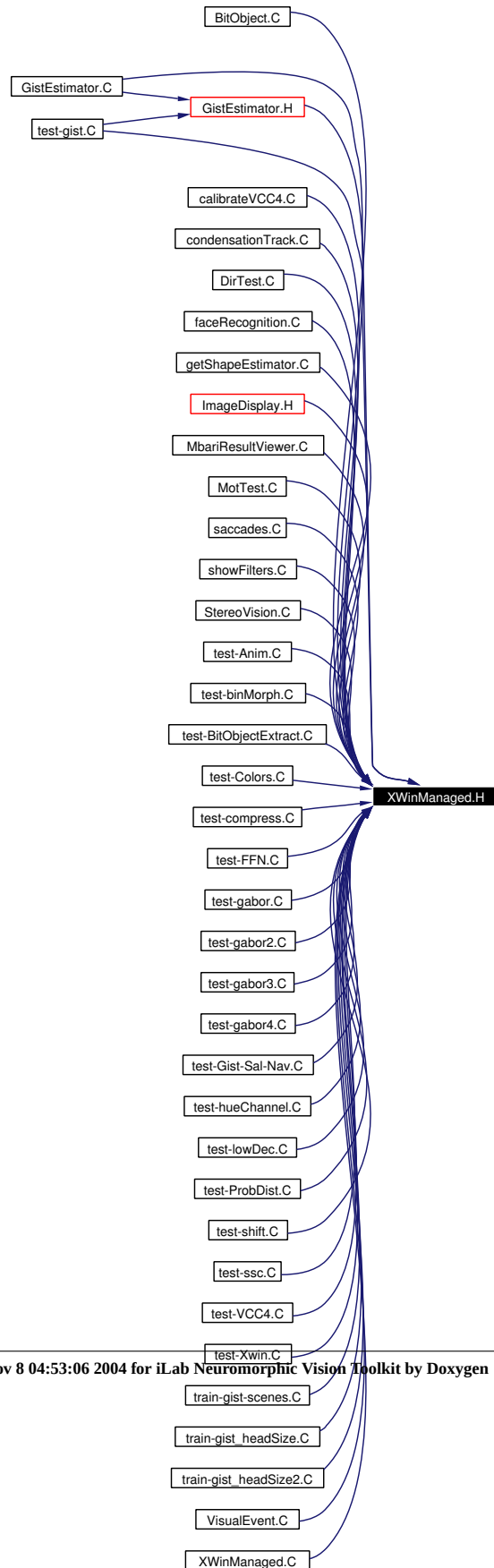
## 11.554 XWinManaged.H File Reference

```
#include "XWindow.H"  
#include <pthread.h>  
#include <X11/Xlib.h>  
#include <vector>  
#include <algorithm>  
#include <unistd.h>  
#include <deque>
```

Include dependency graph for XWinManaged.H:



This graph shows which files directly or indirectly include this file:



## Classes

- class [XWinManaged](#)  
*A window class with active window management.*
- class [CloseButtonListener](#)  
*watch for close button actions with all registered [XWinManaged](#) windows*

## Typedefs

- typedef void `*(thread_function_t)(void *)`

### 11.554.1 Detailed Description

A window class with active window management

Definition in file [XWinManaged.H](#).

## Chapter 12

# iLab Neuromorphic Vision Toolkit Page Documentation

### 12.1 Building models using the toolkit

#### Overview

A number of facilities are provided to allow easy creation of new models. These off-load most of the work involved in having tunable persistent parameters in your models and in being able to generate and parse command-line options in your executables.

To allow this, a computational model is defined as a managed collection of [ModelComponent](#) derivative objects. These objects are instantiated as a hierarchical collection that varies from model to model and may also vary within one model through runtime selection of certain components or behaviors. [ModelComponent](#) provides a basic facility for building such hierarchies. In addition, it provides facilities for holding a number of [ModelParam](#) data members. These are persistent parameters, in the sense that their values can be saved to a file and later re-loaded from that file (using a [ParamMap](#) structure). [ModelParam](#) objects typically are private to a model component and the hierarchical file format used to store them ensures that no conflict between identical parameters across several components should occur. Finally, [ModelComponent](#) objects may request that some command-line option be created for some of their [ModelParam](#) members, so that the [ModelParam](#) values may be set via the command-line. Because typically those options may be shared among a variety of [ModelComponents](#) (e.g., radius of the focus of attention, or verbosity level), there is a single namespace for all the [ModelParam](#) names that may become valid command-line options (see [ModelOptionDefs.C](#)). So, in order to be exportable as a command-line option, a [ModelParam](#) should have its name registered in a global list of all possible command-line options. When the command-line is parsed, all [ModelParam](#) objects with matching name will be

updated to the value supplied in the command-line.

## The bottom-up view

The following classes are available to implement the above functionality.

- [ModelParamBase](#) is a base class (that cannot be instantiated directly because it has some virtual methods) for a persistent model parameter. At the base level, such parameter is just defined by its name and the base does not yet hold any data. However, it provides an interface for name access, registration of the parameter with its parent component, and other functionality.
- [ModelParam<T>](#) is a template class that derives from [ModelParamBase](#). It adds the actual parameter data as a single instance of a value of type T. When the parameter is saved to a [ParamMap](#), it will appear as a "name value" pair in the file.
- [ModelComponent](#) is a base class that facilitates building hierarchies of objects that may hold [ModelParam<T>](#) parameters, that may request command-line options, and that may have other [ModelComponent](#) derivatives as sub-components.
- [ModelManager](#) is a class that facilitates loading/saving of [ModelParam](#) values to a config file, run-time generation of a set of command-line options that depends on which options have been requested by the particular collection of [ModelComponent](#) objects currently instantiated, parsing of the command-line options and automatic update of the [ModelParam<T>](#) contents for those parameters that have been associated with the option, and saving of config files (plus some added functionality).
- [ModelOptionDefs.C](#) contains a global, shared list of all possible command-line options that may be used by any model in our source tree. When [ModelComponent](#) objects request an option to be added to the command-line, it must exist in this list. There are at least three reasons for having such global shared list rather than having each component fully specify all the characteristics of some command-line option it may wish to instantiate:
  1. it is easier to ensure that two option names will not be used in two different parts of the codebase with different meanings
  2. it allows several unrelated components to easily share the value specified at the command-line (both components just need to request the correct option name; in comparison, realizing that two options are equal although they have been fully defined in two different objects would require that all meta-information (name, long option name, short option name, description, valid values) match. This would make updates a nightmare (e.g., updating



the description of an option that has been used by 25 different components defined in 25 separate source files)).

3. When you wish to create a new command-line option, it would be nearly impossible to check whether a similar one does not already exist if all the definitions were spread out all over the code.

Below we walk through a typical use of these facilities to build and run a computational model that is composed of a run-time selectable collection of [ModelComponent](#) derivatives.

## The top-down view and building models

Typically, building a new model entails the following sequence of actions (for a fuller yet simple example, see `test-audioGrab.C`):

- start a `main()` function

```
int main(const int argc, const char **argv)
{
```

- instantiate a [ModelManager](#) object. It will be in charge of managing a collection of [ModelComponent](#) objects, of loading/saving their persistent data, and of creating and parsing the command-line options depending on the requests of the current collection of components:

```
ModelManager manager("My cool model");
```

- Instantiate a hierarchy of [ModelComponent](#) objects. The natural way to do that in this framework is to immediately attach the objects to a [SharedPtr](#). Typically, each [ModelComponent](#) object will hold a number of [ModelParam](#) persistent data members. In addition, [ModelComponent](#) may request that certain command-line options be added to the command-line. This is done by calling `ModelManager::requestOption(this, param)` where the name of the param must correspond to one of the possible command-line option names defined in the global list in [ModelOptionDefs.C](#):

```
SharedPtr<MyComponent> mycomp(new MyComponent(manager));
```

- Register the top-level components with the manager. Upon load/save of parameter files, the manager will invoke the corresponding load/save functions on all its registered components. Those functions are recursive. so that each component will invoke the same function in all of its subcomponents:

```
manager.registerComponent(mycomp);
```

- Once you have registered your hierarchy of components, you may optionally decide which of their internal parameters you wish to export as command-line options; you may do that in batch style using (see [ModelOptionDefs.H](#) for possible values to use here):

```
// Note: this is optional; if you don't do this call explicitly, it
// will be made automatically (exporting ALL options) just before
// we start parsing the command line:
manager.exportOptions(OPTEXP_ALL);
```

- In addition, if you wish to explicitly export a [ModelParam](#) from one of your components as a command-line option (assuming the component has not already done so), you can do that using [ModelComponent::doRequestOption\(\)](#) here (this is a recursive call so that it can also turn on option export in deeply nested sub-components that you may not have direct access to):

```
mycomp->doRequestOption("MyParamName", true, true, true);
```

- Parse the command-line, using the [ModelManager](#) built-in mechanism. This will update the values of all the requested model options, depending on which ones are specified on the command-line. The min/max number of expected non-option arguments are passed, along with a text description of what those are, to `parseCommandLine()`. If an incorrect number of arguments is found, or bogus options are specified, `parseCommandLine()` will print a usage message and return false. There are several special command-line options that are implemented by the manager itself, including `-debug`, `-test-mode`, `-load-config-from=file` and `-save-config-to=file`. If the `-load-config-from` option is encountered during parsing, the config file is loaded and all [ModelParam](#) values are updated immediately. So any command-line option specified before the `-load-config-file` is likely to be overwritten by the values in the config file. Any command-line option specified afterwards, however, will be applied after the config file has been loaded. `-save-config-to` has no immediate effect. Rather, one should call `manager::saveConfig()` at some appropriate time (for example, after a successful run of the model), which will automatically save the config file if the option was specified, or be a no-op if the option was not specified. Finally, one last feature is that the manager may attempt to automatically load a config file named `"~/executablename"` just before parsing the first option, if it was instructed to do so at construction (by default, all three of these features are on. See the [ModelManager](#) constructor for details):

```
if (manager.parseCommandLine(argc, argv, "<input.ppm> <output.ppm>",
                             2, 2) == false) return(1);
```

- Get the extra non-option arguments:

```
int numargs = manager.numExtraArgs();
std::string infilename = manager.getExtraArg(0);
std::string outfilename = manager.getExtraArg(1);
```

- If any additional configuration needs to be made now that the command-line options have been parsed, they should be made here. This may for instance include instantiation of components whose nature is defined at run-time through some command-line option.
- The configuration may be automatically saved if `--save-config-to` was specified on the command-line (the call below will be a no-op otherwise). The call below is optional, and will be made as part of `start()` below if it has not been made explicitly:

```
// Note: This call is optional too, and if you don't make it
// explicitly, it will be made from within start():
manager.saveConfig();
```

- Start all the components through a single recursive call. Typically each component will then perform any additional setup that could not be done in its constructor because it depended on command-line or persistent parameter values:

```
manager.start();
```

- Use the model and get some interesting results. Once the model has started, all functions that modify `ModelParam<T>` values will throw a fatal error if used, so that external people cannot mess with the guts of a [ModelComponent](#) once it has started.
- If you need to revert your model to its initial state (e.g., your program runs three independent simulations between the `start()` and `stop()` calls), try a:

```
manager.reset();
```

to reset your model. But check that your various model components indeed do implement `reset()`, as `reset()` has been implemented only on a few components to date (since most programs run only one simulation and hence to not need any `reset()` capability).

- Once done with a simulation, proper termination of the executable starts by recursively stopping all components.

```
manager.stop();
```

- We are now ready for an exit that will destroy the manager and the hierarchy of components.

```
return 0;
}
```

## 12.2 Changes from Revisions 3.0 to 3.1 in src3/

### Overview of changes in src3/ with Revision 3.1

- **Towards a neural simulation environment**
  - *->Towards a neural simulation environment*
- **Tips and frequently asked questions**
  - *->Tips and frequently asked questions*

## 12.3 Towards a neural simulation environment

Over the many years of development of this toolkit by many independent developers, several bad things have been growing to intolerable proportions:

- only one executable, `vision.C`, had complex command-line parsing capability, although in principle many other executables could have had such capability through the `OptionSet` and `OptionParser` classes. Thus, `vision.C` had become the only program that could be tuned to do various things using command-line options, and as such people would keep adding new stuff into `vision.C` to the point that it had become an incredibly complicated mess;
- many little programs were developed to test out and use new classes added to the toolkit. Typically those programs all shared substantial components (e.g., get a bottom-up visual attention model going, which meant instantiate a bunch of `ChannelBase` derivative objects, a `VisualCortex`, a `Brain`, etc), which was achieved through cut-and-paste from similar programs. Consequently, any core modification of the toolkit required updating of all those duplications of code.
- people had a tendency to develop undocumented classes in their little corner, and the lack of doc and availability to core programs would make this unknown to other developers; thus, several almost-equivalent classes had been created for, e.g., access to a serial port and other functionality.

With release 3.1 of the toolkit in early 2003, we attempted to fix those problems through the following strategy:

- a mechanism was implemented to create tunable model parameters, whose values could be easily set via the command line or via config files. A set of classes (`ModelParam`, `ModelComponent`, `ModelManager`) was provided to ease the management of hierarchies of model components that would have such tunable params, and to automate the process of tuning those parameters via the command line or config files.
- thus, `vision.C` was obsoleted and replaced by `ezvision.C`, which is a truly minimalistic program. Instead of doing all the work of parsing the command-line and deciding what to do based on the command-line options in the `main()` of various programs, that work was shifted `main()` to where it belonged in the first place: the various components of a model. The main paradigm shift, hence, was that a given model component (e.g., `VisualCortex`) should itself declare which of its parameters could be tuned, rather than the `main()` of `vision.C` or other executables deciding on that. With this new framework, adding a new tunable parameter to a given model component would automatically make it available to all programs that used that component, without any modification necessary on those programs' `main()` function.

- [ModelComponent](#) was developed to be a base class for substantial computational modules of a neuromorphic model. The functionalities included in the base class (mechanisms to build hierarchies of [ModelComponent](#) objects, automatic setting of [ModelParam](#) values via the command-line, etc) would not need to be duplicated anymore as it had been the case previously.
- Hopefully, documenting code and making it available to other programs should become more common in this framework too; indeed, if you wish to introduce new functionality into [ezvision.C](#), typically you should not do so by adding code to [ezvision.C](#) itself, but rather by adding new components or new parameters to the existing components that are used by [ezvision.C](#).

Thus, the main idea behind revision 3.1 was to push as much code as possible out of `main()` and back to where it truly belonged in the first place, scattered across the various components of a model. See [here](#) for a quick overview of how this was implemented and how to use this framework to build your new classes and models.

## 12.4 Tips and frequently asked questions

**How do command-line options work?** For a general introduction without which the details below may be difficult to understand, see [here](#).

Command-line options use a shared namespace and are all defined in [ModelOptionDefs.C](#). See [here](#) for an explanation of why that is. The general behavior is as follows: only `ModelParam<T>` data members from [ModelComponent](#) objects may be exported as command-line options; each [ModelComponent](#) requests to the [ModelManager](#) that some or all of its `ModelParam<T>` data members be exported to the command-line; the [ModelManager](#) is in charge of parsing the command-line, and will set the `ModelParam<T>` value to the parsed command-line value for each [ModelComponent](#) that requested the corresponding command-line option. The process is entirely dynamic: if no [ModelComponent](#) requested a given option, it will not be available (and will not show up in the `-help` page); if parsing an option and setting the corresponding `ModelParam<T>` values triggers new options to be exported, they will become available immediately (see below for more detailed explanations of how to add command-line options that depend upon previously parsed command-line option values).

Typically, things go as follows:

- as [ModelComponent](#) derivative objects are instantiated, they instantiate any `ModelParam<T>` data member they may have. By default, those members remain private to the component and its derivatives (thus, they usually are in the protected section of the class definition). If the component is not registered with the [ModelManager](#) using [ModelManager::registerComponent\(\)](#), then the Manager will not interact with the component at all. If the component is registered, the [ModelParam](#) becomes accessible through calls to [ModelComponent::setModelParamVal\(\)](#), [ModelComponent::setModelParamString\(\)](#), etc. run on the manager object (see [ModelComponent.H](#)), since the [ModelManager](#) is a [ModelComponent](#) whose registered components are SubComponents. So, once a [ModelComponent](#) has been registered, you may not need to keep a pointer to it around, as you can modify its parameters through accesses to the [ModelManager](#). This is the way to change [ModelParam](#) values for those parameters which are not command-line options. See below for the parameters that will also become tunable via the command-line.
- some time before the command-line is parsed, one typically calls [ModelComponent::exportOptions\(\)](#) on the manager; this will propagate down to all its registered components and their subcomponents. When this is called, the caller, who has some meta-knowledge about what other components are available in the overall model, chooses which classes of command-line options should be exported (e.g., all options that have to do with saving results; see the `OPTEXP_XXX` defines in [ModelOptionDefs.H](#)). Typically, in [ModelComponent::exportOptions\(\)](#), components decide on what they wish to export and then make appropriate calls to [ModelManager::requestOption\(\)](#) on the manager, to actually

request each option. If `exportOptions()` has not been called by the time the command-line is parsed, it will be called automatically with `OPTEXP_ALL`.

- The manager keeps a list of default values for all possible options, and, for each option, a list of all the components that have requested it. When the manager is instantiated, the default values are taken from the hard-coded list in [ModelOptionDefs.C](#). When a component requests an option for one of its `ModelParam<T>` members, by default the parameter value will be set to the default value from the manager. Subsequent calls to [ModelManager::setOptionValString\(\)](#) on the manager will change that default value as well as change the [ModelParam](#) values of all the components that have requested the option.
- Thus, it does not matter whether you call [ModelManager::setOptionValString\(\)](#) before or after your component is instantiated (if it exists, the [ModelParam](#) value will be changed immediately; if it does not exist yet, the [ModelParam](#) value will be changed when the component is instantiated and calls [ModelManager::requestOption\(\)](#) on the manager).
- Do not use [ModelComponent::setModelParamVal\(\)](#) on the manager to change option values! This will change the values of the corresponding [ModelParam](#) members of all components currently registered with the manager, but it will not change the default option value; so any new component will still get the old default value rather than the value you have just set.
- During parsing of the command-line, [ModelManager::setOptionValString\(\)](#) is internally called onto the manager, for each parsed command-line option. This will change the default in the manager and the internal value of all the registered components, using [ModelComponent::setModelParamString\(\)](#) on each registered component and its subcomponents. If you need to do something each time one of your [ModelParam](#) values is changed, you can overload [ModelComponent::setModelParamString\(\)](#). This is for example used by [OrientationChannel](#) in [Channels.C](#) (to instantiate a bunch of `GarborChannels` each time the number of orientations is changed) and by [SaccadeControllerConfigurator](#) in [SaccadeControllers.C](#) (to instantiate a saccade controller of appropriate type and export its command-line options each time a saccade controller type is specified). With this mechanism, you can thus create new command-line options on the basis of the values of previous command-line options. This is exactly what [SaccadeControllerConfigurator](#) is about. Compare the list of options related to saccade controllers when you type `"ezvision -help"` and `"ezvision -sc-type=Monkey2 -help"`
- To debug the process, simply use `"-debug"` or `"-save-config-to=debug.pmap"` and check whether the value specified on the command-line was properly propagated down to all components that requested it. Otherwise check that you are following the sequence described here and [there](#) in your model. You can also



use a call to "manager.printout(std::cerr)" to print out the model hierarchy and [ModelParam](#) values.

**What if I want to hide options from the user except for a couple?** If you need to be more specific than the general export classes defined for [ModelComponent::exportOptions\(\)](#) with OPTEXP\_XXX in [ModelOptionDefs.H](#), you will need to manually request the options using [ModelComponent::doRequestOption\(\)](#) on your [ModelComponent](#) objects.

An example of this is provided in the implementation of [RadioDecoder::exportOptions\(\)](#) in [RadioDecoder.C](#). Radiodecoder has an [AudioGrabber](#) subcomponent; we would like to let the users choose the device file name for the grabber, but we don't want them to change the audio recording frequency, otherwise the radio decoding will not work. So we explicitly block recursion in the call to [exportOptions\(\)](#) at the level of the [RadioDecoder](#), and manually export the device name option of the [AudioGrabber](#) subcomponent.

Another possible approach is to make a very conservative call to [ModelComponent::exportOptions\(\)](#), for example, exporting nothing, and then to explicitly export some select options by calling [ModelComponent::doRequestOption\(\)](#), on the manager (which will propagate to all components of the model), or on some model components.

**How do I use my own custom option values instead of those provided in [ModelOptionDefs.C](#)?** There are several ways of doing that. You can change an option value using [ModelManager::setOptionValString\(\)](#) before you parse the command line, to set a new default value. See how this is used in [bmcvfigs.C](#) for an example.

Another approach is that you may want to change a bunch of default values depending on the type of derivation of an object which you will instantiate on a given run of your model. Furthermore, in such case, typically the object type initially is not known, and becomes known only while parsing the command-line. An example of that is for [FrameGrabber](#) objects; we would like to set the defaults that are most likely to work for either a [V4Lgrabber](#) or an [IEEE1394grabber](#), but both sets of defaults differ substantially. To solve this, first, we use a [FrameGrabberConfigurator](#) to select the type of [FrameGrabber](#) derivative to use (see definition in [FrameGrabber.H](#)). Then, in [FrameGrabber::exportOptions\(\)](#), we instruct the [ModelManager](#) to use our current [ModelParam](#) values as option defaults, rather than the global defaults in [ModelOptionDefs.C](#); this is achieved by setting the third argument of the [ModelManager::requestOption\(\)](#) calls to true (it is false by default). Finally, in the constructors of [V4Lgrabber](#) (see [V4Lgrabber.C](#)) and [IEEE1394grabber](#) (see [IEEE1394grabber.C](#)), we set our custom defaults. So as a [V4Lgrabber](#) or an [IEEE1394grabber](#) gets instantiated and gets a chance to [exportOptions\(\)](#), the correct defaults are being pushed into the [ModelManager](#). Be careful with this, typically you would want to use this technique only if you expect to have only one object of the type in question in your model.

**Ok, that sounds cool, but what if I have several instances of a given object but want to have different options for each instance?** In this kind of case, you typically will set the parameters that differentiate your objects by hand, and then will export by hand only those options which may be shared by your objects.

An example of that is in `test-stereo.C`, which uses two [IEEE1394grabber](#) objects, for the left and right eyes. The first grabber is manually configured to use [FrameGrabber](#) subchannel 0 using [ModelComponent::setModelParamVal\(\)](#), and the second one to use subchannel 1. Clearly, it hence does not make sense for this setup to export a command-line option to set the subchannel. So instead, we export no option through the standard [ModelManager::exportOptions\(\)](#) call. Then we only export a few options like image size and grab mode, by hand, calling [ModelComponent::doRequestOption\(\)](#) on the manager. This will recurse and both grabbers will request the option since we have registered both with the manager, and when command-line options are parsed, they will affect both grabbers equally.

### How do I replace one of my regular internal parameters by a `ModelParam<T>`?

- First, make sure than you can create a [ModelParam](#) of the type of your parameter. For that, check the list of instantiations of `ModelParam<T>` in [InstantiateAll.H](#). If your parameter type is in this list, move on to the next step. Otherwise:
  - Enter a new instantiation in the list in [InstantiateAll.H](#), and `#include` in [ModelParam.C](#) (towards top) whatever file is required to define the type you are using, so that the instantiation will work as [ModelParam.C](#) is compiled;
  - If operator`>>` and operator`<<` exist for your new type, move on to the next step;
  - If that type is an enum, create a `.H` file for it, and model it after the Pyramid-Type enum in [PyramidTypes.H](#); make sure you number your enum values and provide the `xxxName()` function;
  - We need to be able to convert to/from string for your new parameter type; to this end, edit [StringConversions.H](#) and add prototypes for [convertToString\(\)](#) and [convertFromString\(\)](#) specializations for your new type. Then implement those functions in [StringConversions.C](#); look at how the other conversions were implemented and just do the same for your type.
  - If you are going to export your new type as a command-line option, edit [ModelOptionDefs.H](#) and add a key for your new type in the `ModelOption-Type` enum.
- At this point, 'make `ModelParam.o`' and 'make `StringConversions.o`' should work.
- Add the new data member in the protected section of your class that will use the parameter; your class must derive from [ModelComponent](#) or one of its deriva-

tives. Make sure you put doxygen comments as to what the [ModelParam](#) data member does; see for example the protected section of class [Brain](#) in [Brain.H](#).

- In your constructor for your class, add an initialization for your parameter, with a default value. See the constructor in [Brain.C](#) for an example.
- If you want to export your new parameter as a command-line option, you will need to implement an overload of the [ModelComponent::exportOptions\(\)](#) function. Decide when you want to export the option, depending on how much it relies on the presence of other components in the model; look in [ModelOptionDefs.H](#) at the `OPTEXP_XXX` defines. Typically, models that will only use your component will call `exportOptions()` with only `OPTEXP_CORE` on. Then implement `exportOptions()` for your new class, looking for example at the implementation for class [Brain](#) in [Brain.C](#). Make sure your overload calls the base [ModelComponent::exportOptions\(\)](#) at some point. Finally, create a new entry in [ModelOptionDefs.C](#) for your new option. The name for your [ModelParam](#) should be chosen so that:
  - if it is unlikely to be used by other components, prefix it with something that has to do with your class, so that you will be sure that nobody will use the same name by mistake (e.g., "AudioGrabberStereo" rather than "Stereo");
  - if you expect that several components will share the value set by your option, then use a more generic name (e.g., "FOAradius"). Look at the top for the format of the entries in the `AllModelOptions` array in [ModelOptionDefs.C](#). Choose an option category for your option; this will place your option along with related options when `-help` is requested.
- Everything should now compile and models using your class will all benefit from the presence of the new [ModelParam](#) and possible associated command-line option.

**How do I set the FOA radius?** That's a good one. Once you have called [ModelComponent::start\(\)](#) on your manager, you can't change option values anymore (that's to prevent people from changing, e.g., the number of orientations in an [OrientationChannel](#) while a simulation is running). But you need to start the model and load the first input image in order to get its dims and compute the FOA radius. We solve this in two ways:

- the preferred way is to just use an [InputFrameSeries](#) to load your input frames (see [FrameSeries.H](#)). The [InputFrameSeries](#) will peek the dims of the first frame during `start()`, and set a few options that are dependent on these dims, like `FOAradius`, `FoveaRadius`, etc, if those currently are set to zero (which means that they should be set from input image dims).

- or there is the manual way, but it is non-preferred. You can just use a raw [Raster::ReadRGB\(\)](#) call to read your first frame before you call `start()` and after you have parsed the command-line. Then you manually compute your FOA radius, then set the option by calling `setOptionValString()` on the manager (you will need to convert your radius to string, and can use [convertToString\(\)](#) from [StringConversions.H](#) to do that). Then start your model.

## 12.5 Changes from src2 to src3

### Overview of changes in src3

- **The make/build process**
  - →*separate compilation and faster builds ( $\sim 1.3\times$  full, 5-15x partial)*
- **Test suite(s)**
  - →*automated testing and bugs caught faster*
- **Ref-counting and copy-on-write**
  - →*clearer syntax and  $\sim 1.5\times$  faster code using [Image](#)*
- **Image iterators**
  - →*STL style and STL algorithms, efficiency, safety with "checked iterators"*
- **Free functions for image operations**
  - →*syntax and encapsulation*
- **C++ standard library**
  - →*[The std:: namespace](#), [Use of std::numeric\\_limits](#), [STL](#)*

## 12.6 The make/build process

### One Makefile

- don't need separate `MakeObjects` or `Makefile.defs` files; their previous contents are now part of the one `Makefile`
- don't need recursive `make` calls

### One dependency file

- don't need separate `.dep` files
- all dependencies go into the `alldepends` file
- a Python script generates `alldepends` **quickly** (< 1 sec), by being conservative (ignores `#ifdef`'s), by ignoring system include files, and by parsing each file only once

```
# ... Makefile ...

alldepends: $(SOURCES) $(HEADERS)
    time ./cdeps.py ./ --objdir '' > $@

include alldepends
```

### Explicit template instantiation

- Allows separate compilation of `.C` files into `.O` files
- Don't have to `#include` the `.C` files into each executable

```
// ... Image.C ...

// Generate code for all visible function definitions:

#ifdef INST_BYTE
template class Image<byte>;
template class Image< PixRGB<byte> >;
#endif
#ifdef INST_FLOAT
template class Image<float>;
template class Image< PixRGB<float> >;
#endif
```

### g++ version 3.0.x

- Upgraded version offers much-improved library support for the 1998 ANSI/ISO C++ standard
- more details [later](#)

**Advantages of separate compilation**

- Shorter full build times:
  - assume `make clean`, then for full build:
  - `make all` in `src2` takes 241 CPU seconds
  - `make all` in `src3` takes 184 CPU seconds
  - ==> **1.3x faster**
- Much shorter partial build times:
  - assume full build, followed by `touch Image.C`, then for partial re-build:
  - `make all` in `src2` takes 175 CPU seconds
  - `make all` in `src3` takes 12 CPU seconds
  - ==> **15x faster!**
  - **-OR-**
  - assume full build, followed by `touch vision.C`, then for partial re-build:
  - `make all` in `src2` takes 21 CPU seconds
  - `make all` in `src3` takes 4 CPU seconds
  - ==> **5x faster!**

## 12.7 Ref-counting and copy-on-write

How it works, in four parts:

- [Dims](#) represents the width and height dimensions:

```
struct Dims
{
    int32 const ww; // the width
    int32 const hh; // the height
};
```

- [ArrayData](#) encapsulates the management of a 2-D data array

```
template <class T>
class ArrayData
{
    long        itsRefCount; // reference count, is managed by ArrayHandle
    Dims const itsDims;      // the width and height, fixed at construction
    T* const itsData;        // the data array, fixed at construction

public:
    // ... constructors/destructor omitted

    const T* data() const { return itsData; } // const version
    T* dataw()           { return itsData; } // non-const version

    ArrayData* clone() const { return new ArrayData(*this); }

    void incrRefCount() const { ++itsRefCount; }

    long decrRefCount() const { return (--itsRefCount); }

    bool isShared() const    { return (itsRefCount > 1); }
};
```

- [ArrayHandle](#) offers ref-counted, copy-on-write access to [ArrayData](#)

```
template <class T>
class ArrayHandle
{
    ArrayData<T>* px; // pointer to our data

public:
    // ... constructors/destructor omitted

    // REF-COUNTING:

    ArrayHandle(const ArrayHandle& other) :
        px(other.px)
    {
        px->incrRefCount(); // just share other's data, and bump the ref-count
    }
};
```



```

~ArrayHandle()
{
    if (px->decrRefCount() == 0) // drop the ref-count, and if we were the
                                // last user, then delete the ArrayData
        delete px;
}

// COPY-ON-WRITE:

const ArrayData<T>& get() const
{
    return *px;                // const version just returns the ArrayData
}

ArrayData<T>& uniq()
{
    if (px->isShared())          // non-const version checks if the ArrayData
                                // is shared, and makes a fresh copy if needed
    {
        *this = ArrayHandle(px->clone());
    }

    return *px;
}
};

```

- [Image](#) wraps an [ArrayHandle](#), and offers an expanded interface:

```

template <class T>
class Image
{
    ArrayHandle<T> itsHdl;

    const ArrayData<T>& impl() const { return itsHdl.get(); }
    ArrayData<T>& uniq()             { return itsHdl.uniq(); }

public:
    int32 getWidth() const { return impl().w(); }
    int32 getHeight() const { return impl().h(); }

    void setVal(int32 index, const T value)
    {
        uniq().dataw()[index] = value;
    }
};

```

### Syntax

- Since copying is cheap, an `Image<T>` can be the return value of a function...

```

template <class T> void coolFilter(const Image<T>& src, Image<T>& result);

template <class T>

```

```
Image<T> hotFilter(const Image<T>& src);

template <class T>
void foo(const Image<T>& x)
{
    Image<T> coolResult;
    coolFilter(x, coolResult);

    Image<T> hotResult = hotFilter(x);
}
```

- ...and chaining operations is easier:

```
template <class T>
void foo(const Image<T>& x)
{
    Image<T> tmpResult, coolResult;
    coolFilter1(x, tmpResult);
    coolFilter2(tmpResult, coolResult);

    Image<T> hotResult = hotFilter2(hotFilter1(x));
}
```

### Avoid copies

- We only have to deep copy the data when absolutely necessary. For instance, if we are converting an `Image<T>` of unknown type to `Image<byte>`:

```
template <class T>
void process(const Image<T>& x)
{
    Image<byte> xbyte = x; // without ref-counting, this always does a deep copy
                          // with ref-counting, avoids copying if T is byte
}
```

### Reduce bugs

- Since memory management is encapsulated within `ArrayData` and `ArrayHandle`, "ordinary" functions should not have to call `new[]/delete[]` to manage temporary storage

```
template <class T>
void Image<T>::coolDraw()
{
    T* buf = new T[w*h]; // get temporary storage

    // ... fill in buf ...

    T* old = a;          // clean up
    a = buf;
    delete [] old;
```

```

}

template <class T>
void Image<T>::hotDraw()
{
    Image<T> tmp;

    // ... fill in tmp ...

    this->swap(tmp);
    // or
    *this = tmp;
}

```

### Exception safety

- if an exception is thrown while filling in buf in coolDraw(), then we leak memory
- but if an exception is thrown while filling in tmp in hotDraw(), then ~Image() will clean up tmp for us

### Efficiency

- to run test suite [before]: 17.3 CPU seconds
- to run test suite [after]: 12.8 CPU seconds
- ==> *1.35x faster*
- to run CINNICtest on simp.circle.pgm [src2]: 46 CPU seconds
- to run CINNICtest on simp.circle.pgm [src3]: 44 CPU seconds
- ==> *1.05x faster*

### "Gotchas"

- non-const functions must check uniqueness each time they are called
- in particular, it is not especially efficient to call `Image::setVal()` in the middle of some inner loop, since
  - `Image::setVal()` must check `Image::coordsOk()` each time it is called, and
  - `Image::setVal()` must check uniqueness each time it is called
- therefore, in time-critical loops, instead of this:

```
Image<float> a;  
for (int y = 0; y < a.getHeight(); ++y)  
  for (int x = 0; x < a.getWidth(); ++x)  
    a.setVal(x,y,42);
```

- try this if you like `while` loops:

```
Image<float>::iterator itr = a.beginw(), stop = a.endw();  
while (itr != stop)  
  *itr++ = 42;
```

- or this if you like `for` loops:

```
for(Image<float>::iterator itr = a.beginw(), stop = a.endw();  
    itr != stop;  
    ++itr)  
{  
  *itr = 42;  
}
```

## 12.8 Image iterators

### Quick iterator review

- Iterators act like pointers
- All standard library containers provide iterators

```
#include <vector>
#include <iostream>

void foo()
{
    std::vector<int> myvec;

    // fill in with some data
    for (int i = 0; i < 10; ++i)
        myvec.push_back(i);

    // print out the contents using iterators
    for (std::vector<int>::const_iterator
         itr = myvec.begin(),
         stop = myvec.end();
         itr != stop;
         ++itr)
    {
        std::cout << *itr << std::endl;
    }
}
```

- Iterators can be used with generic algorithms:

```
template <class Iterator>
void printContents(Iterator itr, Iterator stop)
{
    while (itr != stop)
        std::cout << *itr++ << std::endl;
}

void foo()
{
    std::vector<int> myvec;

    // ... fill in with some data ...

    // print the contents
    printContents(myvec.begin(), myvec.end());
}
```

### Image iterators: normal and debug versions:

- Normal (non-debug) iterators are just pointers:

```

template <class T>
class Image
{
public:
#ifdef MEM_DEBUG

    typedef      T*      iterator;
    typedef const T* const_iterator;

    const_iterator begin() const { return impl().data(); }
    const_iterator end()   const { return impl().end(); }

            iterator beginw()      { return uniq().dataw(); }
            iterator endw()        { return uniq().endw(); }
#endif
};

```

- Debug iterators are objects that act like pointers, but check that the pointer is in bounds every time it is dereferenced.
- To use debug iterators, first make `clean`, then add `-DMEM_DEBUG` to `CFLAGS` in the Makefile, then make `all`.

```

template <class T>
class Image
{
public:
#ifdef MEM_DEBUG
    // ... normal iterators ...
#else

    template <class T>
    class ck_iter                    // a checked iterator class
    {
    TT* ptr;
    TT* start;
    TT* stop;

    // ... ++, --, <, >, <=, >=, +=, -=, all those goodies ...

    TT* operator->() const
    {
        ImageHelper::ck_range(ptr, start, stop);
        return ptr;
    }
    TT& operator*() const
    {
        ImageHelper::ck_range(ptr, start, stop);
        return *ptr;
    }
    TT& operator[](diff_t d) const
    {
        ImageHelper::ck_range(ptr+d, start, stop);
        return ptr[d];
    }
    };

```

```

typedef ck_iter<T>          iterator;
typedef ck_iter<const T> const_iterator;

const_iterator begin() const
{ return const_iterator(impl().data(), impl().end()); }

const_iterator end() const
{ return const_iterator(impl().end(), impl().end()); }

iterator beginw() { return iterator(uniq().dataw(), uniq().endw()); }
iterator endw()   { return iterator(uniq().endw(), uniq().endw()); }

#endif
}

```

- Iterators in use with handmade loops:

```

template<class T>
void Image<T>::clear(const T& val)
{
    for (Image<T>::iterator itr = this->beginw(), stop = this->endw();
         itr != stop; ++itr)
    {
        *itr = val;
    }
}

```

- Iterators in use with STL algorithms:

```

#include <algorithm> // for std::count(), etc.
#include <numeric>   // for std::accumulate(), etc.

template <class T>
int32 Image<T>::emptyArea() const
{
    return std::count(this->begin(), this->end(), T());
}

template<class T>
double Image<T>::getMean() const
{
    double sum = std::accumulate(this->begin(), this->end(), 0.0);

    return sum / double(getSize());
}

```

## 12.9 Free functions for image operations

- Some [Image](#) member functions have become free functions, defined in separate .H files. Why do this?
  - Since all operations can be coded in terms of iterators, most [Image](#) algorithms do not need access to the `private` parts of [Image](#).
  - In some cases free functions offer a more natural mathematical syntax than member functions.
  - Keeping unrelated functions in separate files improves logical encapsulation and decreases compile-time dependencies.

| File name                          | Contents                                                                                                                                    |
|------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------|
| <a href="#">Image_All.H</a>        | <code>#includes</code> all other Image-related files                                                                                        |
| <a href="#">Image_ColorOps.H</a>   | color operations: <code>get/set</code> components, luminance, RGB-YIQ conversions, etc.                                                     |
| <a href="#">Image_FilterOps.H</a>  | <code>lowPass*()</code> , <code>convolve*()</code> , <code>sepFilter*()</code> , etc.                                                       |
| <a href="#">Image_MathOps.H</a>    | <code>absDiff()</code> , <code>average()</code> , <code>takeMax()</code> , <code>RMSerr()</code> , etc.                                     |
| <a href="#">Image_Omni.H</a>       | all omni-directional related operations                                                                                                     |
| <a href="#">Image_ShapeOps.H</a>   | <code>rescale()</code> , <code>downSize()</code> , <code>concat*()</code> , <code>dec*()</code> , <code>int*()</code> , <code>crop()</code> |
| <a href="#">Image_Transforms.H</a> | <code>segmentObject()</code> , <code>dct()</code> , <code>infoFFT()</code> , etc.                                                           |

### New syntax examples

```

void oldFoo(const Image<float>& x)
{
    Image<float> filtered = x;
    filtered.lowPass3(true, true);
}

#include "Image_FilterOps.H"

void newFoo(const Image<float>& x)
{
    Image<float> filtered = lowPass3(x, true, true);
}

void oldFoo(const Image<PixRGB<byte> >& x)
{
    Image<float> lum;
    x.luminance(lum);
  
```



```
}

#include "Image_ColorOps.H"

void newFoo(const Image<PixRGB<byte> >& x)
{
    Image<float> lum = luminance(x);
}

void oldFoo(const Image<byte>& x)
{
    Image<float> smaller = x;
    smaller.rescale(x.getWidth()/2, x.getHeight()/2);
}

#include "Image_ShapeOps.H"

void newFoo(const Image<byte>& x)
{
    Image<float> smaller = rescale(x, x.getWidth()/2, x.getHeight()/2);
}
```

## 12.10 Test suite(s)

- Black-box tests: test the external behavior of an executable
  - So far, there is a test suite for the bin/vision executable that exercises all of its command-line options and compares the output against verified results
- White-box tests: test the (internal) behavior of a class and its methods
  - So far, there is a small test suite for core [Image](#) functionality, e.g.:

```
void Image_xx_type_convert_xx_1(TestScaffold& scaffold)
{
    float array[4] = { -10.9, 3.2, 254.7, 267.3 };

    Image<float> f(array, 4, 1);

    Image<byte> b = f;

    REQUIRE_EQ((int)b.getVal(0), 246); // convert to unsigned and overflow
    REQUIRE_EQ((int)b.getVal(1), 3);
    REQUIRE_EQ((int)b.getVal(2), 254);
    REQUIRE_EQ((int)b.getVal(3), 11); // because of overflow
}
```

- The whole thing is automated, and can be run with `make test` from the main `src3` directory, which runs `tests/run_test_suite.tcl`.

```
#!/usr/bin/env tclsh

foreach f [glob test_*.tcl] {
    if { [file executable $f] } {
        eval exec ./$f $argv >&@ stdout
    }
}
```

- Please add to the test suite! Write a script that matches the pattern `test_*.tcl` and put it in `src3/tests`, and it will automatically be included in the test suite.
- This way people can run `make test` before checking new changes into the repository to make sure that nothing has been inadvertently broken.

## 12.11 The std:: namespace

### "New-style" header files and the std:: namespace

- old: `#include <iostream.h>`, contents go into global namespace

```
#include <iostream.h>

int main()
{
    cout << "Hello, World!" << endl;
}
```

- new: `#include <iostream>`, contents go into std namespace

```
#include <iostream>

int main()
{
    std::cout << "Hello, World!" << std::endl;
}
```

- **-OR-** *using declarations*

```
#include <iostream>

using std::cout;
using std::endl;

int main()
{
    cout << "Hello, World!" << endl;
}
```

- **-OR-** *using directive*

```
#include <iostream>

using namespace std;

int main()
{
    cout << "Hello, World!" << endl;
}
```

- **-BUT- Never put using directives or declarations into a header file!**

- C library headers: old-style

```
#include <math.h>
#include <stdlib.h>
#include <string.h>
```

- C library headers: new-style

```
#include <cmath>
#include <cstdlib>
#include <cstring>
```

## 12.12 Use of std::numeric\_limits

std::numeric\_limits

- old:

```
#define BYTEMIN 0
#define BYTEMAX 255
#define INT16MIN (-32768)
#define INT16MAX 32767
#define INT32MIN (-2147483647)
#define INT32MAX 2147483647
#define FLOATMIN -3.40e+38F
#define FLOATMAX 3.40282347e+38F

void foo(Image<float>& x)
{
    x.normalize(BYTEMIN, BYTEMAX);
}
```

- new:

```
#include <limits>

void foo(Image<float>& x)
{
    x.normalize(std::numeric_limits<byte>::max(),
               std::numeric_limits<byte>::min());
}
```

- Why? It's standard, it's portable, it offers more
  - *can be specialized for user-defined types*
  - get the number of bits used to represent the type
  - get the range of the exponent for floating-point types

```
#include <limits>

template <class T>
void foo(const Image<T>& x)
{
    if (std::numeric_limits<T>::is_signed)
    {
        // do something for a signed type
    }
    else
    {
        // do something for an unsigned type
    }
}

template <class T>
```

```
void bar(const Image<T>& x)
{
    if (std::numeric_limits<T>::is_integer)
    {
        // do something for an integral type
    }
    else
    {
        // do something for a floating-point type
    }
}
```