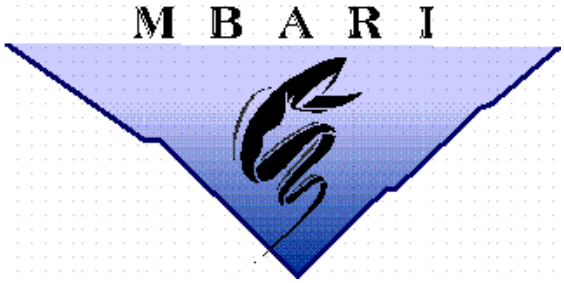
The MBARI Automated Video Event Detection Project



Automated Video Event Detection System

Software Requirements Specification

Version 0.6

Revision History

0.1 15 Jun 2004 - DLD Revised initial draft
0.2 15 Sep 2004 - DLD Revised draft
0.3 May 3, 2005 – DEC Revised event detection parameters table, added video input and buffering, user interface, and data and display storage matrix. revised references.
0.4 May 5, 2005 – DEC Revised real-time process flow diagram, expanded real-time-case, modified data display and storage table, modified event tracking, VCR control sections. Added section 3.4.4.
0.5 May 12, 2005 – DEC Revised real-time process flow diagram, added use scenario descriptions and classifier section.
0.6 Thursday, May 19, 2005 – DEC. Added event monitor section, added user case overview, updated all diagrams

Automated Event Detection Software
Requirements Specification

Figure and Tables

1. Introduction

The purpose of the automated video event detection system is to provide a system based on neuromorphic methods that will automatically locate objects (events) on a video stream that would ordinarily be noticed as an object of interest by a human user. The objective of the neuromorphic algorithms are to simulate, as closely as possible, the detection of objects that would capture the attention of a human user, by simulating the actual neural mechanisms in the eye and brain that autonomically govern attention This system can then be used to pre-process video footage as an aid to identify objects of interest and as an aid to further annotation of these events.

The current prototype technology can be used to identify several objects in priority order of interest, as defined by a saliency-based attention guided algorithm developed by collaborators at Caltech and USC. The purpose of this document is to identify the software requirements that will guide the MBARI design and development of the software system based on this technology, from its prototype form through its form as a stand-alone application or as a component of the Video Annotation and Retrieval (VARS) system (http://tienhou.shore.mbari.org/vimsproject/).

The first phase of requirements, concept design and implementation of a prototype is to identify the most suitable prototype features of the application as well as identify future potential uses, enabling us to develop a concept design for a prototype implementation for development. A secondary task of the analysis will be to determine how this application could interface with the VARS system.

As a minimum, the prototype will be able to create a stream of data that can be assimilated by users of a variety of video systems as well as the VARS system to aid in post-cruise video annotation. A long-term goal will be to integrate the application for real-time use with the VARS system during cruise operations.

1.1. Purpose

The purpose of this document is to identify the high-level system requirements that must be elaborated by the initial software prototype. A basic principle of early prototyping is that delivery of some operational capabilities early in the development process permits incorporation of new requirements and capabilities that did not exist or were not feasible at the start of the development. There are general questions that must be answered and documented before each phase of the software development process can be initiated. Such questions include a) What is the software supposed to do? B) How does the software interact with people, the system's hardware, other hardware and other software? c) What are the performance requirements, speed, response time, and recovery time in the case of system failure? d) What are the attributes of the system including maintainability and portability of the software, security, etc. and e) what constraints must be met including power, memory size, operating system environments, implementing language, etc.

This document is intended not only for the benefit of the designers and implementers of the software prototype but also for users of the system, as well as other designers and developers of other systems (such as VARS) that may utilize this software as a subsystem.

1.2. Scope

These requirements are confined to the automated video detection system. It does not address requirements of any other system except as they may relate to this system. In particular the VARS requirements are described in other documents which may be referenced here.

It is not the purpose of this specification document to discuss or describe how specific requirements are to be met, but rather describe the requirements of users as well as software requirements that need to be met for the application to fulfill its role as prototype system. There are, however, some cases where non-functional requirements are specified, due to MBARI convention or constraints due to the saliency-based software interface requirements.

1.3. Acronyms

The following is a list of acronyms provided for the reader's convenience:

COTS	Commercial Off The Shelf software
OOP	Object Oriented Programming
VARS	Video Annotation and Reference System
AVED	Automated Video Event Detection project or process
TBD	To Be Determined
UTC	Coordinated Universal Time
LTC	Longitudinal Timecode
RTC	Real-time Clock
OS	Operating System
PPM	Portable Pixel Map
PNG	Portable Network Graphics
API	Application Programming Interface
SDI	Serial Digital Interface
HD-SDI	High Definition Serial Digital Interface
NAS	Network Appliance Server

1.4. References

The primary documents that are referred to in this document:

1) IEEE Std 830-1998 Documentation Standard, specifically the section on *Recommended Practice for Software Requirements Specifications*. This is the reference guide that describes the form of this document.

2) "Requirements Engineering", Merlin Dorfman, *SEI Interactive*, 03/99,

Requirements documents for the VARS system -http://tienhou.shore.mbari.org/vimsproject/

1.5. Overview

The sections below begin with an overall description of the factors that led to the requirements for an automated video event detection tool.

The first section describes the context of the video annotation process as it is used at MBARI. There is a description of the user characteristics as well as the software and hardware interfaces. There is no discussion in this overall description of any specific subsystem requirements.

This is followed by a section containing all of the automated video event detection user and software requirements to a level of detail sufficient to enable designers to design a prototype system to satisfy those requirements. This section includes specific performance requirements as well as constraints on the design.

2. Overall description

A goal of this effort is to apply the powerful neuromorphic saliency model of visual attention (*NRN 2001*) that has been implemented in software by Itti *et al.* (*PAMI 1998*), a processing approach inspired by biological vision systems, to detect visual events in imagery collected by MBARI systems. We augment the saliency-based model with selected best approaches from machine vision work on object extraction, size estimation from single camera imaging, and efficient implementation in software, including deployment on compute clusters.

2.1. Typical work flow scenarios

In the AVED system there are two types of workflows most use cases fall under: one for batch processing and another for real time use. In the batch processing scenario a user selects a sequence of image frames or sequence of time codes. The frames collected from this sequence, and event detection parameters are then fed into the automated event detection processing system. The AVED system then operates frame-by-frame and outputs event data consisting of images, time codes, and other metadata corresponding to each event. This data can then be displayed by the user as an overlay on the original image sequence. The event data can also be exported for further processing by an annotation system or some other processing system.

The real-time processing scenario is similar to the batch processing scenario, except the event data can be previewed and edited while processing.

2.2. Batch and real-time flow descriptions

A basic flow diagram that shows the movement of the data through the automated event process in the batch processing mode is shown in Figure 1. In the first steps the video must be digitized from the primary imaging source for subsequent analysis by the AVED process. The primary imaging source data and associated metadata consist of digitized video and metadata describing at a minimum the source of the data and actual time it was recorded as a primary key.

The user then selects the sequence of images, or image clips for processing, and the digitized video is buffered to enable flexibility in the processing.

The output of the AVED process is one or more "events", that are primary object locations and/or regions identified as "objects of interest" by the algorithm and associated metadata about those events like maximum size, duration, etc. These events may then be optionally input into the classification process to identify the events.

In the event display and edit process, it is assumed that the event metadata can be displayed in an overlay on the master digital video source.

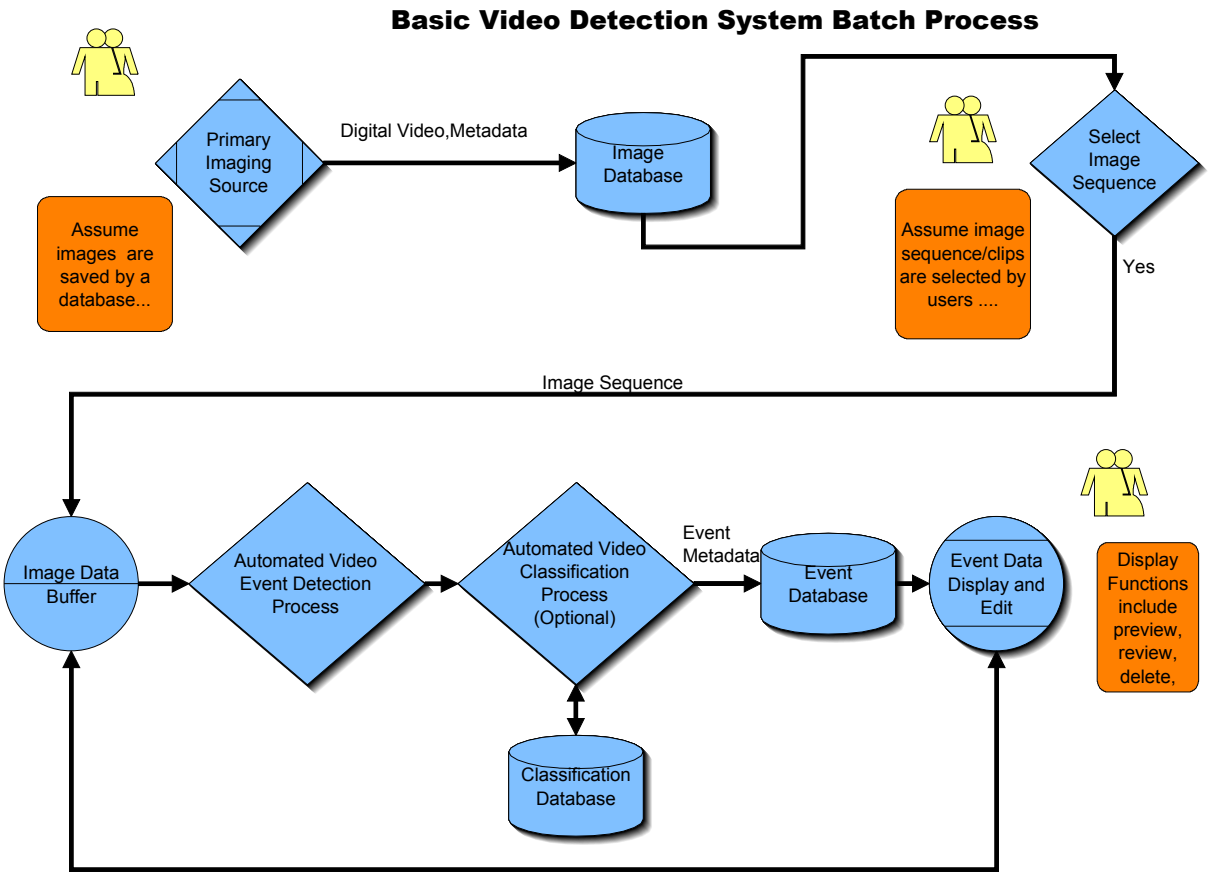


Figure 1. Batch process work flow

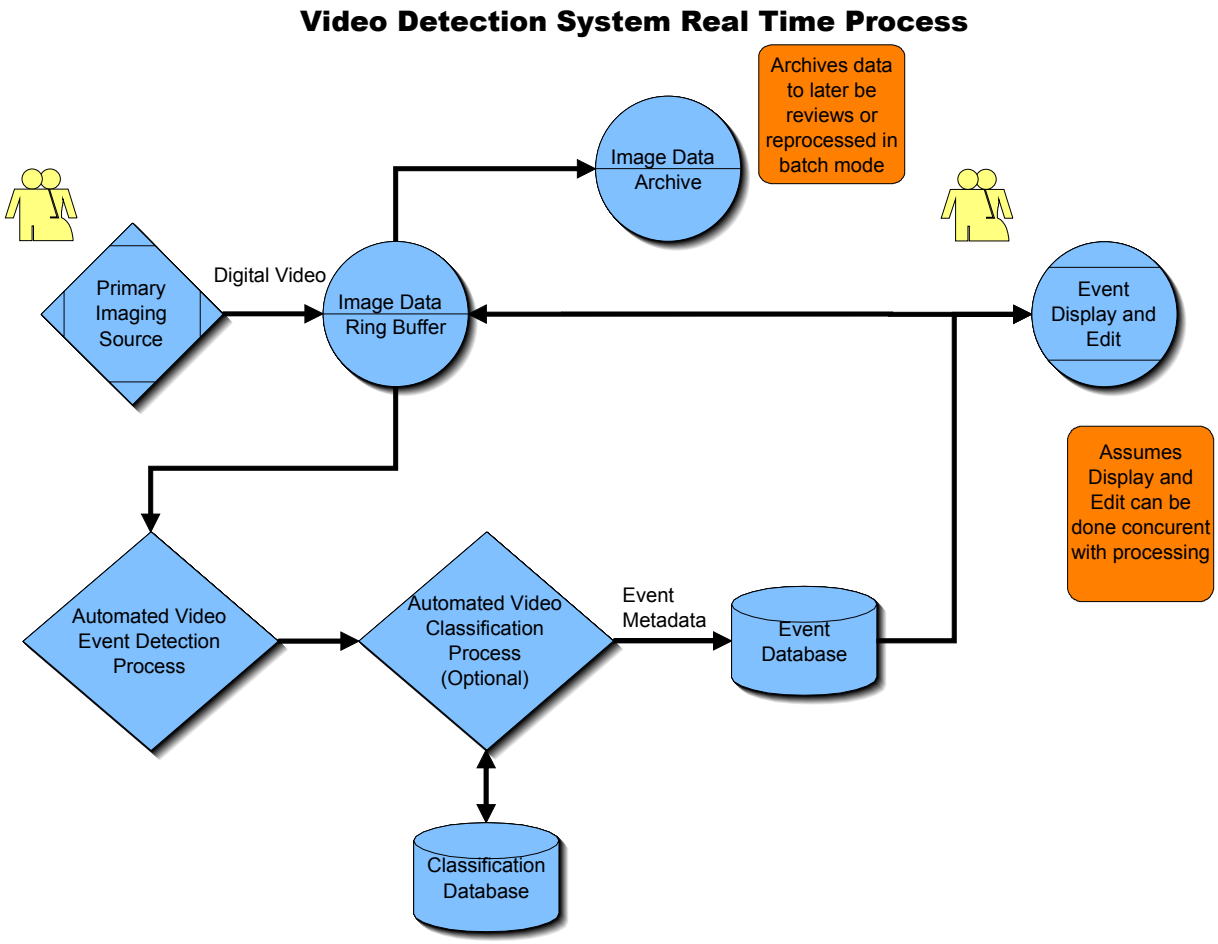


Figure 2. Real-time process work flow

Figure 2 shows the flow diagram of movement of data through the automated event process in a real-time processing mode. This flow could be used, for example, when the primary imaging source is an observatory, or a ROV video camera. This real-time process differs from the batch mode, in that event display can be done while AVED processing. This real-time system contains several software modules designed for completion of real-time video analyses.

2.3. Use cases

The following section describes the possible use cases for AVED software. These cases are not comprehensive, but are the best understood cases at this time. These cases are included to give an overview of how a user may use an AVED system, and more importantly, to provide the basis from which the detailed requirements are derived from.

Figure 3. AVED Use Cases shows a general model of the system user requirements. This figure illustrates the different users and use cases. It also shows some system behaviors like the ability to process both real-time and video clips that are transparent to most users, but are a required system capability that may be of general interest to the reader.

The AVED system provides two core requirements: the ability to find interesting events in a sequence of frames, and the ability to classify these interesting events. How, and in what combination these capabilities are used depends on the case.

The three potential users of this system are the MBARI annotator, the science user, or the public user.

For the annotator, it is the AVED classifier that could offer the most useful capability. It could be used, for example, to assist in finding targeted animals during normal annotation sessions, leaving the annotator time to focus on other annotations. It could also be used to find animals that were annotated incorrectly, or missed entirely on previously annotated dives.

Science users may use it in conjunction with an observatory, for example, by setting up event monitors to look for specific animals, events, or sequences of events (e.g. to capture a spawning event) that might otherwise be difficult to capture. AVED could then send notification to the user when this event trigger occurs.

Finally, the public user may use AVED technology transparently when simply viewing a smart" video stream from an observatory video server that was sent key frames of "interesting" events by the AVED algorithm.

These different use cases will be discussed in the following sections.

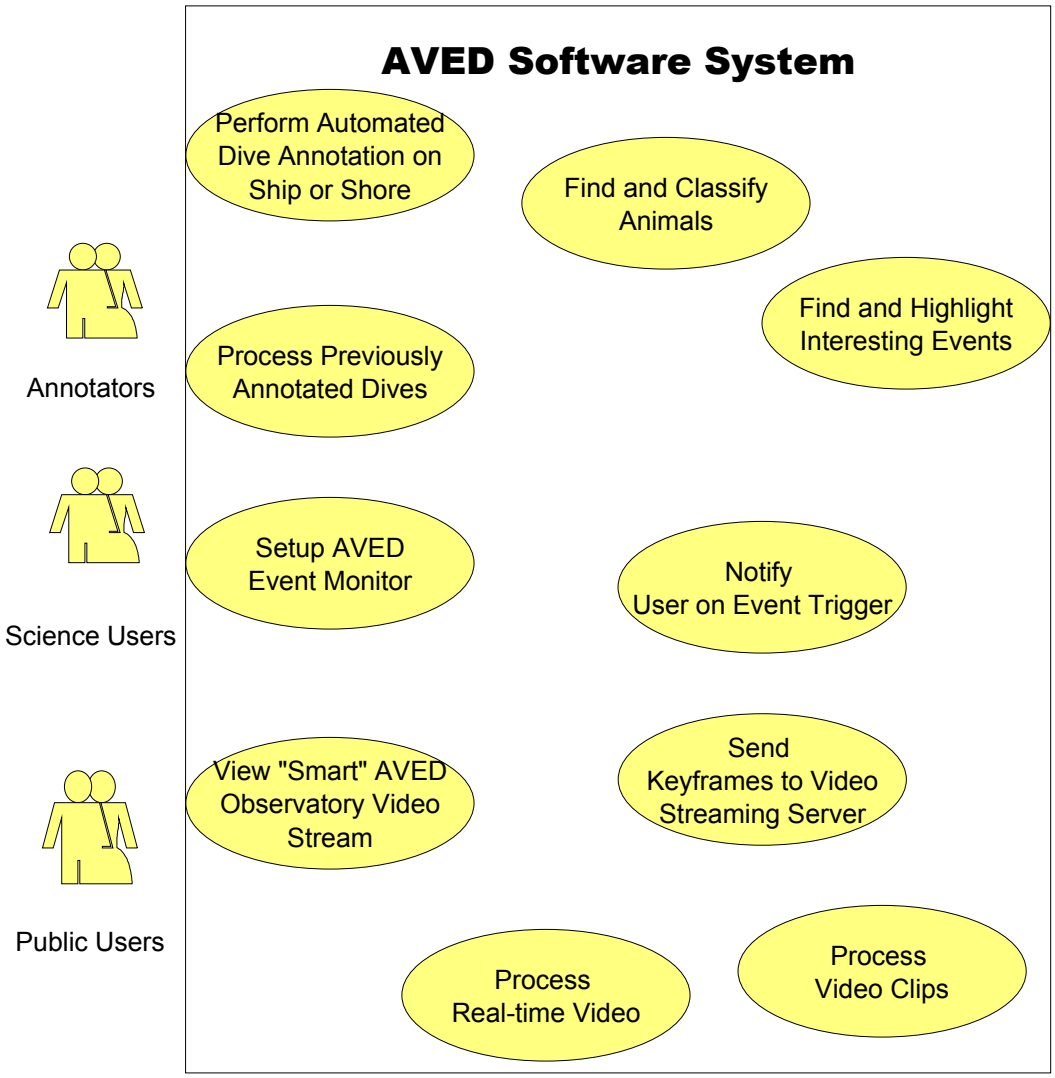


Figure 3. AVED Use Cases

2.3.1. Using AVED as an Automated Dive Annotation Assistant

AVED could be used as an Automated Dive Annotation Assistant. Several cases stem from using AVED as an Annotation Assisant. One case would be helping the video lab annotator by automating the search for common or easily identifiable animals. These animals could then be ignored during the normal annotation session. In another case the science technician could use the Annotation Assistant to automate the search for targeted species for science research..
In all cases, the AVED system looks much the same and would improve the annotator's productivity by assisting in some current function and optimally improve their existing workflows.

It is assumed in both cases, results would be merged into VARS database, and the database would be later queried for analyses. It is also assumed, that if the user wanted to preview and edit the results, they could do so either through the AVED or VARS

interface. Otherwise, the results would be merged by the annotator into the existing video database following an annotation session.

The following diagrams illustrate the use case and data flow in this case.

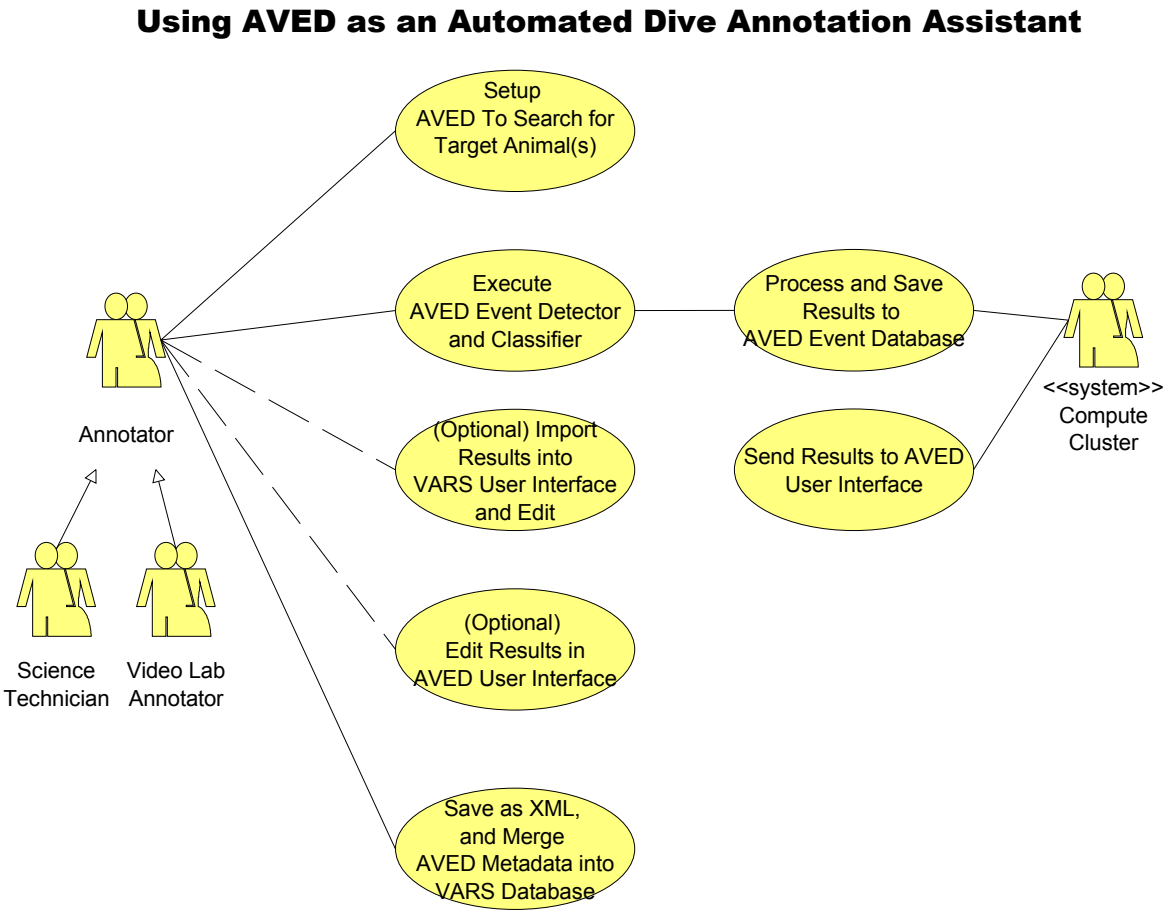


Figure 4. AVED Automated Dive Annotation Assistant Use Case

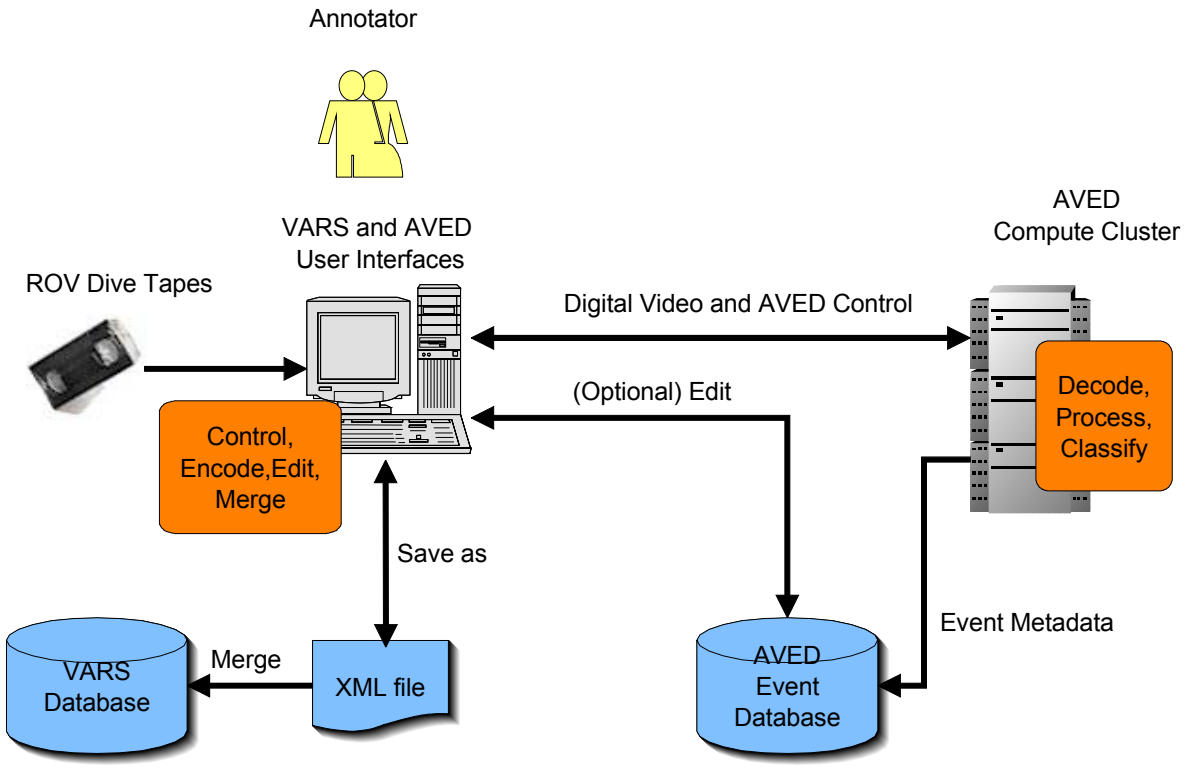


Figure 5. AVED Automated Dive Annotation Assistant Data Flow

2.3.2. Using AVED to reprocess previously annotated dives

AVED can also be used to reprocess previously annotated dives. This case is a subset of using AVED as an Automated Dive Annotation Assistant, except the results would need to be imported into the VARS user interface and edited to resolve any conflicts with existing annotations. This might be used to find animals that were annotated incorrectly, or missed entirely.

Using AVED to Process Previously Annotated Dives

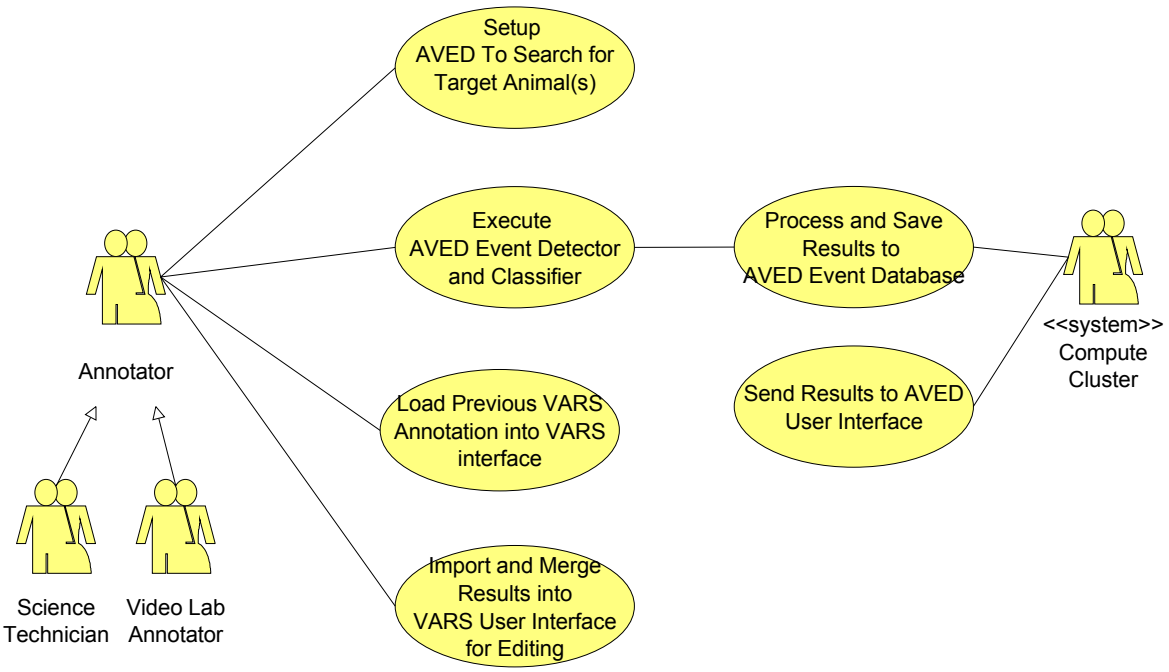


Figure 6. Using AVED to Process Previously Annotated Dives

2.3.3. Using AVED as a Shipboard Annotation Assistant

AVED could also be used as a component of a shipboard video annotation system. This is not much different that the case of using AVED as an Annotation Assistant, except the video input is from a real-time feed versus a tape. The data flow, would be similar to that described by the the real-time work flow in section 2.2.

The real-time digital video could either be processed in real-time, or particular segments could be stored for later reprocessing by selecting IN/OUT points for video archive storage. Results are then (optionally) edited, then saved and merged into the VARS shipboard database.

The following diagrams illustrate the use case and data flow in this case.

Using AVED as Shipboard Annotation Asssistant Tool

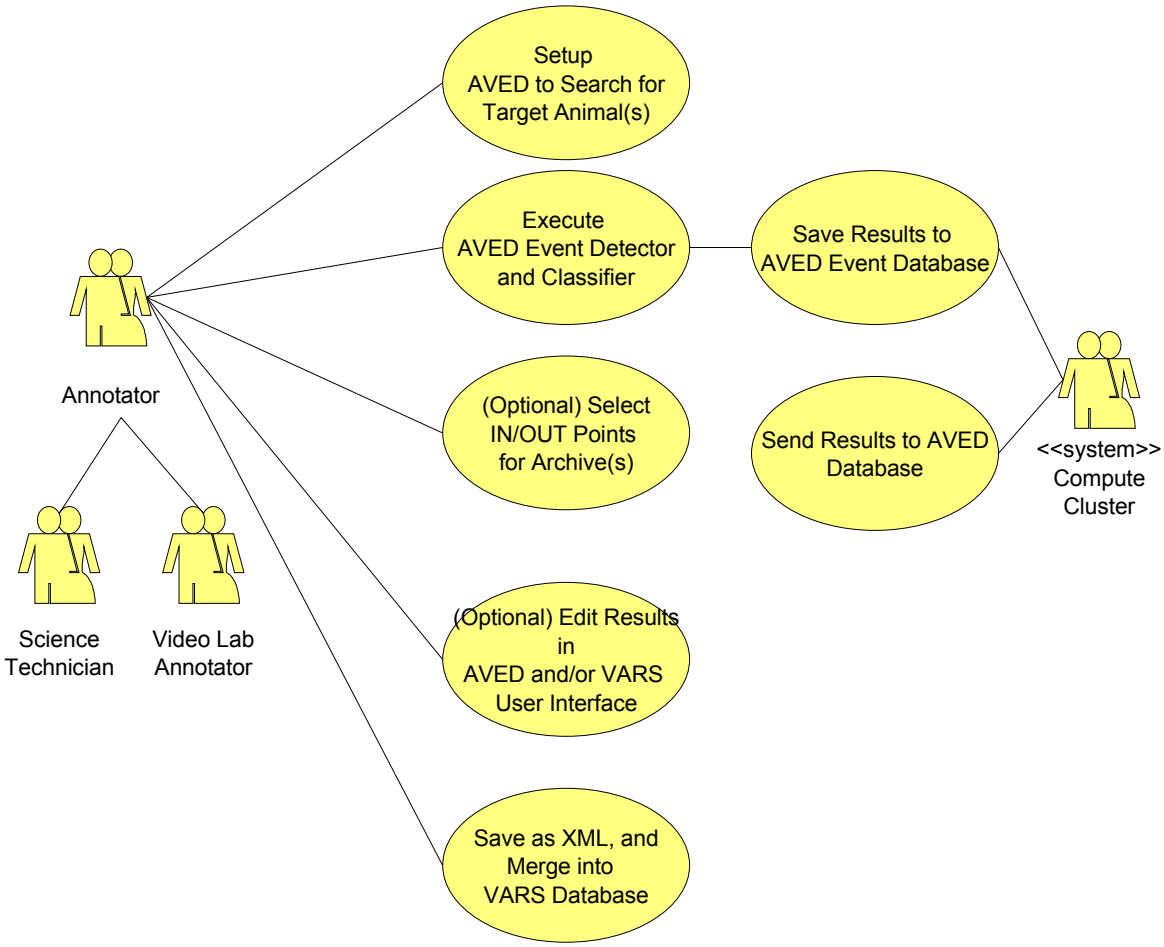


Figure 7. AVED Shipboard Annotation Assistant Use Case

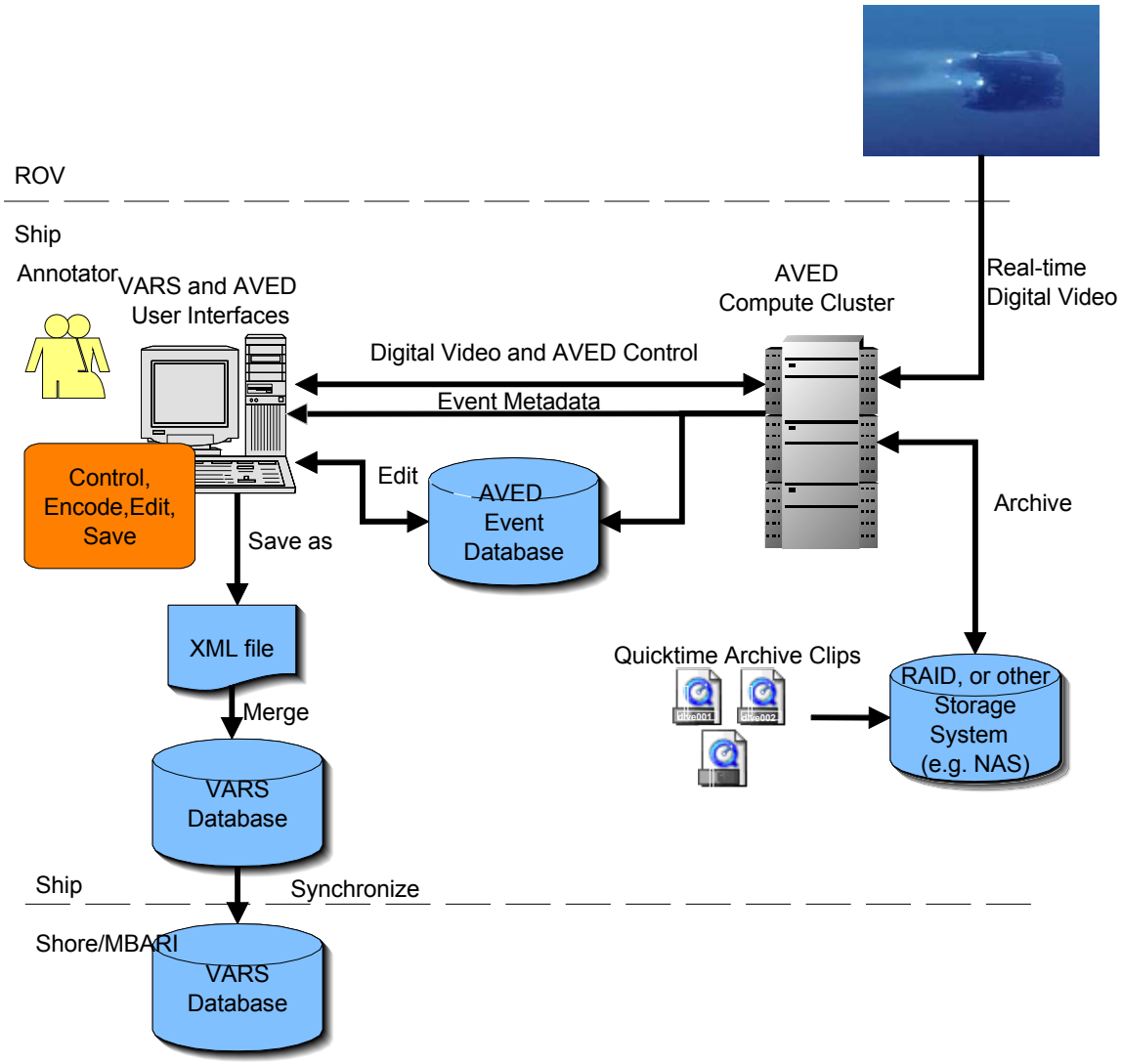


Figure 8. AVED Shipboard Annotation Assistant Tool Data Flow

2.3.4. Using AVED as an Intelligent Observatory Video Filter

The twenty-four hour video streams from observatories offer a unique and challenging case for the AVED software. These video streams require video management system to archive, analyze and generally manage video in a way that is useful to the science community. As an integrated part of a observatory video workflow, AVED could be viewed as a type of intelligent video filter.

As an intelligent filter, AVED could be used to find and highlight interesting events, or automate classifcation of known animals for input into an observatory video annotation database.

In the simplest case, it could be a key-frame generator that sends keyframes when a new "interesting" event is detected by the AVED algorithm to a video streaming server in the observatory chain.

It could also be used by the science users as an video event monitor that would notify a science user when their user-defined event has occurred. This could be useful to monitor, when a spawning event occurs, or to monitor video events associated with a science experiment for example.

The following diagrams illustrate the use case and data flow used in this case.

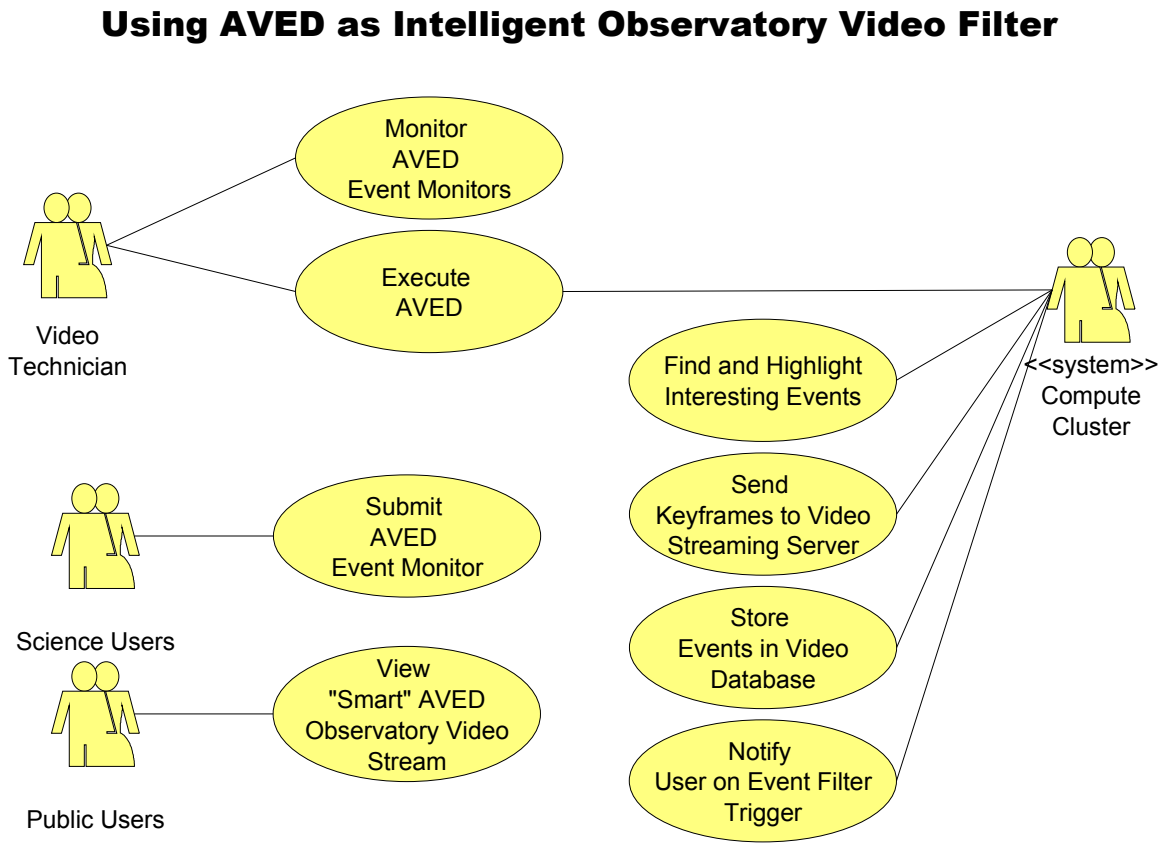


Figure 9. AVED Intelligent Observatory Video Filter Use Case

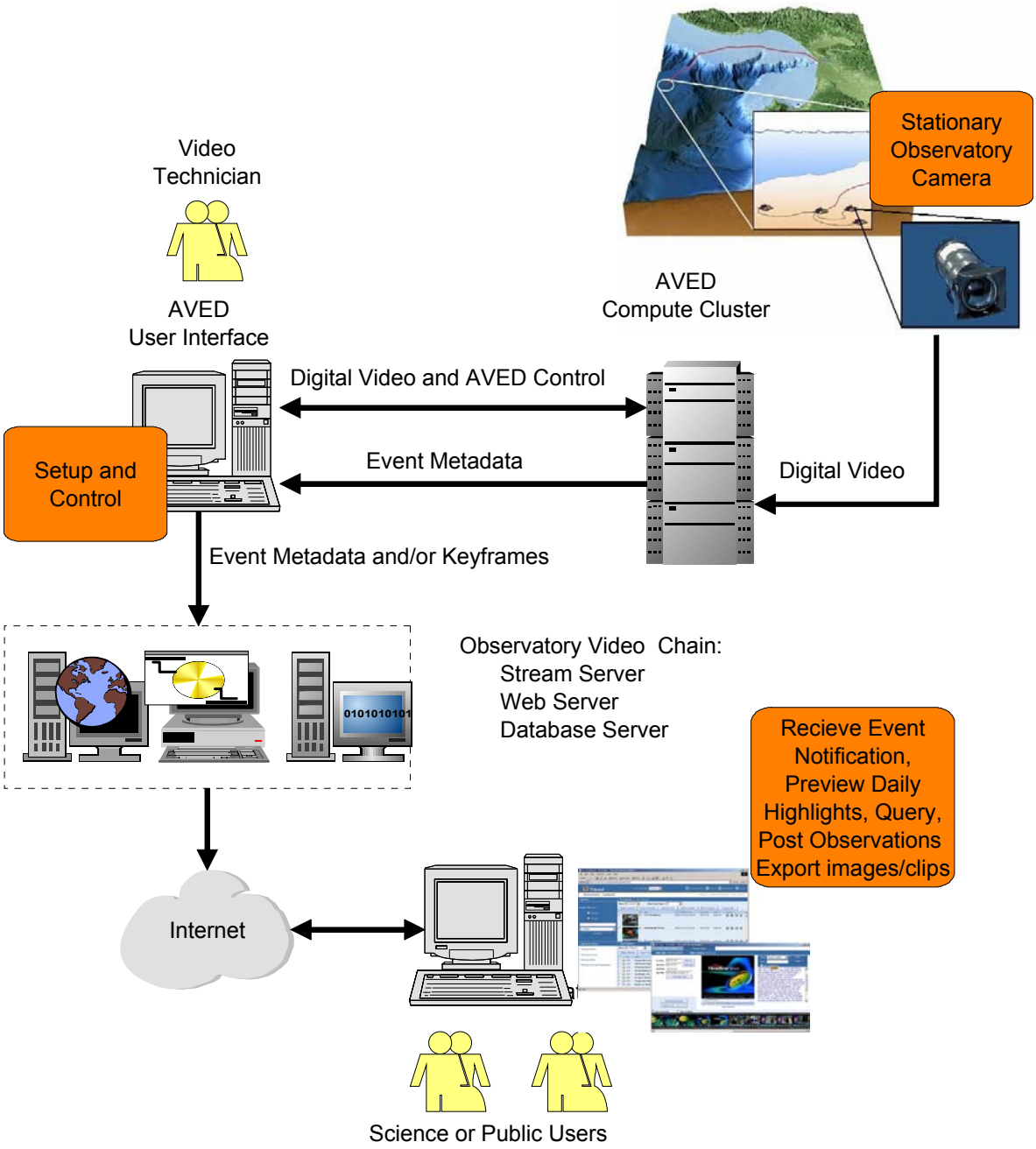


Figure 10. AVED Intelligent Observatory Filter Data Flow

2.4. Constraints

2.4.1. General design

The software is to be designed and implemented in modern, maintainable technology, such as an object oriented programming language (OOP), which is supported over the useful life of the system. Software systems are to be well documented and to include written software test and maintenance procedures. There should also be maximum use of industry proven standards and technology and commercial off the shelf software (COTS).

2.4.2. Computation

The greatest constraint on the system is the computational requirements for executing the saliency-based algorithm on video imagery in real or near-real time. The saliency software is implemented in C++ for use of Linux based systems only.

2.4.3. Lifespan

The project is expected to have a useful operational lifetime of over ten years. Therefore the software must be designed and implemented using tools and technologies that provide a stable development and maintenance environment over this time period.

2.5. Assumptions and dependencies

A primary assumption is that the systems that provide input to the AVED systems will provide the image sequences and required metadata in a format acceptable to the system. Similarly, any processing system that will interface with the AVED system will be capable of accepting the video and metadata output the AVED system provides.

The current assumption is that a specialized compute cluster will be adequate to support the -real time requirements demanded by some AVED users, and the compute cluster or perhaps a high end computing environment is required for batch processing, and that common desktop PCs cannot provide the computing environment required for the AVED system, and that AVED will continue to require a specialized compute cluster for some time. It follows that interfaces to standard desktop computing environments will remain an important part of the AVED design.

3. Specific requirements

3.1. Video input and buffering

The video input to the AVED system must in digitized form for the saliency algorithm. The saliency algorithm processes one frame at a time, so the video input and buffering must support this frame-based system.

3.1.1. Primary imaging sources

The AVED system shall support a primary imaging source of either a QuickTime-encoded 24-bit RGB video file, or a SDI or HD-SDI coded video stream.

Rationale

SDI is a digital format that allows timecode and other video data to be transferred without degradation, making it a good candidate for an imaging input signal. Allowing for a file format like QuickTime provides a convenient way to move around files. This works well for short clips. QuickTime is an open standard format that has freely available encoders on Linux. It is also a format that has cross-platform support for Linux, Windows, and Macintosh computers, making it a good choice for users to capture into this format. This enables a wider community to capture video and put into a format the AVED system can read and process.

3.1.2. Video input format

The digitizing system shall encode video in 24-bit RGB formatted PNG or PPM frames.

Rationale

The saliency algorithm requires data in 24-bit RGB color space. PPM or PNG formats support this RGB format and are supported by the saliency code so using this format leverages the existing code. These formats are also widely supported, documented, and easy to understand frame formats making them a good choice.

3.1.3. Input buffer capacity

A minimum of TBD minutes of data from the primary imaging source shall be stored for processing. This would be provide a TBD minute ring buffer in the case of the real time processing mode, or a TBD minute collection for a batch processing.

Rationale

The amount of storage is not known yet. Different use cases require different buffering capability so further experiments are required to determine this.

3.2. Visual event and tracking definitions

3.2.1. A sequence of objects shall be defined as an event when between three and the *Maximum event duration*, and bounded by the *minimum* and *maximum event area* specified by the user (see **Table 1. Data display and storage matrix** for user parameter descriptions).

3.2.2. When an event exceeds the maximum duration defined by the *Maximum event duration* parameter in **Table 1. Data display and storage matrix**, it shall be reset and tracked as new event

Rationale
This is somewhat arbitrary, but this could make data management easier. All parameters about each event instance are tracked and a long or indefinite event sequence would be difficult to manage

3.2.3. The AVED software shall detect and track between one and TBD visually interesting objects in any given frame in a sequence (see **Table 1. Data display and storage matrix** *Maximum events detected* user parameter description).

Rationale
In video with busy scenes, many objects may need to be tracked at once, but in boring scenes, the user may want to adjust the amount of objects detect.

3.3. Automated event classification

The AVED system shall include automated classification software.

Rational
In addition to event detection, automated classification of events has been identified as a very useful addition to an AVED system. Automated classification of events is useful in cases where common, easily identified objects can be recognized with classification software. Cases where animals are difficult to identify are the subject of ongoing research, and as research continues in this area, improvements may be made to add the ability to classify difficult objects.

A classifier could be used, for example, in an observatory where common species could be identified and displayed for public knowledge. It could also be used to identify and count animals for science study, or to improve MBARI video annotation productivity by classifying common events for the annotator, leaving them time to focus on more difficult annotations.

3.3.1. Classification shall include a graphical tool to create, edit, and generally manage training classes.

Rationale:
The current classifier is trained with a set of images. It is assumed that the training classes will frequently be modified; therefore a tool is required to maintain those images. This tool would allow the user to define new classes, delete classes, or similar functions

as we learn more about what works best. It may also be useful to have this tool as an integrated part of the VARS annotation database.

3.3.2. Training classes shall be defined as a collection of images that represent a particular object.

3.3.3. The classification software shall include the ability to periodically validate an individual class and preview both its correct and misclassification percentages.

Rationale:
The current validation scheme is to test the class by running the classifier on the collection of images in a particular class, against the class itself. The results of the validation include correct and misclassification rates which are indicators of how good a training class is, e.g. a 95% correct classification 5% misclassification would indicate the class is a good discriminate.

3.3.4. The graphical tool shall include the ability to specify the frequency of class validation per individual classes. It shall default to every TBD images by default, otherwise the period shall be defined as either every X number of added images, or as a percent of the total images.

Rationale:
Because some classes will require fewer images than others, it follows that validation of these classes may need to be scaled accordingly. For example, rathbunasters require fewer training images because their features are well defined. However, other small, or not well defined animals like the poeobius require larger training classes before becoming effective discriminators.

3.3.5. The tool shall include the ability to tag and manage versions of individual classes by tracking changes to the class names and class image collections.

Rationale:
It is possible in classification schemes to increase the misclassifications by inadvertently adding poor, or incorrect images to a class. Adding versioning gives the users control to revert to a known good class.

3.3.6. The tool shall include the ability to plot a history of individual class classification and misclassification rates from the validation tests.

Rationale:
This plot would indicate when a class has reached an optimum level. At some point adding images does not improve classification performance and this tool would graphically indicate when that occurs.

3.3.7. The tool and manage versions of individual classes by tracking changes to the collection of images that represent the individual classes.

3.3.8. Output from the classifier shall include a class name, a confidence factor and/or class probability that correlates to an input image.

Rationale:
This confidence factor and/or probability would be an indicator of how well the classifier works. This could be useful for engineering analysis, or simply an indicator the user on how correct the classification is.

3.3.9. Classification user parameters and classifier output shall be stored with the event metadata in the event database.

3.4. Data storage and display

3.4.1. Data listed in**Table 1** shall be stored in a database and optionally, a XML file.

3.4.2. The user may specify the XML output file, and if not specified, an automatic one will be generated based on the UTC day and time and *Input source identification* user parameter (see **Table 1)**.

3.4.3. The following matrix describes the parameters, their adjustable range, and storage and display settings. Parameters noted as *User Parameters* may be adjusted through the AVED user interface (see section 3.6).

Table 1. Data display and storage matrix

Parameter	Range/ Measurement Units	Description	User Parameters
Input source identification	na	ASCII identifier of primary data source, e.g. tape number, diver number, QuickTime filename, video feed, etc.	Y
Output XML filename	na	ASCII filename for event metadata results	Y
Event data		This is event data stored per every frame	N
Event number	0-9999		
LTC timecode	hours:min:secs		
Bounding box	[x1,y1 x2,y2]	Bounding box is upper left and lower right frame coordinates	
Duration	0 – TBD frames**		
Maximum events detected	TBD	Maximum number of points the AVED algorithm finds every 3 frames. The larger this number the more events founds.	Y
Maximum event duration	3 – TBD frames	Range a object can be tracked. Must be tracked for at least 3 frames before labeled a valid event.	Y
Minimum event area	TBD square pixels	Minimum size of object to detect. Generally, the smaller this number, the more events found, including those that are noise, e.g. marine snow.	Y
Maximum event area	TBD square pixels	Maximum size of object to detect. Generally, the larger this number, the fewer events should be found. This number is relative to the size of the input video.	Y
Sliding average cache size	1 – 100	Cache size used for an algorithm that subtracts a sliding average across input frames to remove background objects such as equipment, or other manmade artifacts.	Y

**** This depends on the maximum tracked event duration specified by the user**

3.5. Exchanging data with other applications

3.5.1. The XML format shall be the format used to exchange data with other applications.

3.5.2. The XML formatted output shall be supplied to other applications with an associated XML schema.

3.6. Automated video event user interface

The AVED system shall have a graphical interface to assist the user in running the software and displaying and editing the results.

3.6.1. Setup and control

3.6.1.1. Processing setup

3.6.1.1.1. The user interface shall provide the user the option to select a filename, or video feed.

3.6.1.1.2. The user interface shall provide the ability to select a timecode range or total duration to process.

3.6.1.1.3. The user interface shall provide an option to start or stop processing at any time.

3.6.1.1.4. When processing from a continuous video source, the user shall specify, optionally, the duration to process

3.6.1.1.5. The user interface shall allow the user to save current processing settings into a user-defined profile that can later be reloaded.

Rationale:
The user may want to customize different profiles, e.g. for benthic versus midwater. This allows some flexibility for the users.

3.6.1.1.6. .

Rationale:
This assumes the AVED process can support a real-time capture and process work flow in the case of e.g. an observatory. Currently, the AVED process is not real-time with a 720x486 frame size at 30 frames per second, but may approach real-time with a reduced frame rate and frame size input.

3.6.1.2.Classifier setup

3.6.1.2.1. The user interface shall provide the user the option to turn on/off the classifier.

3.6.1.2.2. The user interface shall allow the user to select what training classes, or collections of classes to test against.

Rationale:
The training classes are what a classifier searches for. The user may want to only search for individual things, or collections of things.

3.6.1.2.3. The user interface shall allow the user to save a collection of classes into a user-defined profile that can later be reloaded.

Rationale:
The user may want to customize different profiles, e.g. for benthic versus midwater. This allows some flexibility for the users.

3.6.1.3.VCR control

3.6.1.3.1. In cases where the primary imaging source is a SDI or HD-SDI video feed from a tape deck, the graphical interface shall provide graphical VCR control to control the video source for processing or preview

3.6.1.3.2. The controls shall include a fast-forward, rewind, play and stop functions. There functions shall operate in the same convention a mechanical fast-forward, rewind, play or stop operates.

3.6.2. The spreadsheet

3.6.2.1.The graphical editor shall provide a spreadsheet style summary of the results where each row represents an event frame.

3.6.2.2.The spreadsheet shall allow the user to collapse any individual event sequence together.

Rationale
This is somewhat arbitrary, but the processed data is frame based and this seems a good fit to represent the saliency algorithm requires data in 24-bit RGB color space. PPM or PNG formats support this RGB format and are supported by the saliency code so using this format leverages the existing code. These formats are also widely supported, documented, and easy to understand frame formats

3.6.2.3. All columns in the spreadsheet shall have variable widths that the user can vary.

3.6.2.4. The spreadsheet shall provide the user the ability to toggle any column, or group of columns, between hidden or shown.

3.6.3. The Event Monitor

An optional component of AVED system shall be a graphical tool to allow users to setup event filters to monitor the event metadata stream.

Rationale:
In the case of an observatory for example, the user may want to setup an event filter to monitor for particular animals, sequences, or some combination that defines a behavior they are interested in.

3.6.3.1. The component shall be used either as a standalone tool, or as an integrated part of AVED

Rationale:
As a standalone tool, this could be useful for the science user to monitor for a spawning event for example, but a fully capable AVED user interface is not needed in this case.

3.6.3.2. This tool shall allow the user to log events to a file.

3.6.3.3. This tool shall allow users to specify a notification script to be called when the event has occurred.

Rationale:
For example, a notification script might be useful if the event of interest was very infrequent and a user just wanted an email of the event and didn't want to bother with a log.. Otherwise, the user could look at event log periodically.

3.6.4. Editing the event results

The user needs to preview the results, and may need to modify the output of the AVED system if the results are incorrect. It is assumed, deleting, modifying, but not adding events may be required.

3.6.4.1. The user interface shall provide the ability to select and delete an event sequence, or continuous sequence of frames of an event and the end or beginning of the event.

Rationale:
This assumes the AVED process may have cases where junk is labeled interesting, and the user may want to filter this out these.

3.6.4.2. The user interface shall provide the ability to combine events together to form one event.

Rationale:
This assumes the AVED process may have cases where a single object is labeled as two separate events. For example, in cases where the animal swims in an out of a field of view, it may appear as more than one separate event, but is actually the same event.

3.6.4.3. The user interface shall provide the ability to add images to any class.

Rationale:
Building classes required "training" them with images. This will initially require adding many images. This provides a way for the user to add images to build these image collections.

3.6.4.4. The user interface shall provide a preview mode to preview each event.

3.6.4.4.1. The user interface shall provide a frame-by-frame, half-speed, or full-speed preview of each event.

3.6.4.4.2. The user shall be able to start this preview, by double-clicking any event in the spreadsheet, or selecting a "preview all" option, that will automate previewing all events.

3.6.4.4.3. In "preview all" mode, the user shall be able to start or stop the preview, at any time.

3.6.5. Saving the event results

3.6.5.1. Event data shall be saved automatically while editing through the user interface.

3.6.6. Video overlay display

In addition to a spreadsheet style user interface, the AVED software shall provide a tool to overlay the event metadata with the data from the image buffer. This may be in the format of an analog or digital video overlay for example.

3.6.6.1. This overlay display must have the best color accuracy possible, and display only the primary full sized video source.

Rationale
The intention of the intention of the overlay is to provide the best interpretation of the results by not changing the quality of the original video with, for example, video compression or resizing. This also generally means that the display must have good contrast and an adjustable gamma setting, or a default gamma setting that displays natural scenery color well.

3.6.6.2. The overlay display shall display a metadata overlay that draws a bounding box around each event.

3.6.6.3. The overlay display shall display all remaining event data in **Table 1. Data display and storage matrix,** that pertains to the frame displayed, in a place that doesn't obscure the event of interest, e.g. the corners of the display.

3.6.6.4. The overlay display shall allow the user to toggle on/off the any display options.

Rationale
The display may interfere with an object, or be too busy for some users, so allowing this to be turned on/off gives the user flexibility to adjust to their preference.

4. Future requirements

4.1. Estimate the size of a non-moving object

4.1.1. Single camera algorithm
4.1.2. Dual camera algorithm

4.2. Estimate the area of total 2D field of view

Appendix A – Use Case interviews –Annotation Assistant

This is a set of notes from a meeting with video lab users held on Aug 20, 2003. The participants were Susan VonThun, Kyra Schlining, Danelle Cline, Dirk Walther, Duane Edgington, Dan Davis, and Alexis Wilson. The meeting began with a presentation by Dirk Walther of the saliency based attention directing features of the technology and how it might be beneficial as an annotation assistant.

Question. (Duane) What sorts of features should an annotation assistant have?

Ans. (Susan) - the ability to fast forward to the next 'interesting' part of the video..

Ans. (Kyra) – be able to be adjusted by the user for what it picks out; "teach the tool".. Also movement is important for annotators.

Note. Kyra and Susan both pointed out that in present practice they *do not annotate* objects unless they can identify them. They do not mark objects for later identification normally.

Ans. (Kyra) – Bounding boxes are not used in VARS but would be useful. Knowing the size of the object would be useful.

Ans. (Susan) – Saving properties of automatically identified objects would be useful (eccentricity, orientation, apparent size etc.)

Question (Duane) Do you plan to some quality control on annotations ?

Ans. (Susan, Kyra) Over the years annotations have been done to different levels of detail, and in some cases, names were later found to be wrong. The annotators know about these problems but outside users would not. There are two areas that affect the quality of annotations: 1) a taxonomic knowledge of objects, and 2) practice and experience in identifying specific objects from specific characteristics of their appearance in video. On-line assistance in either of these would improve the quality of annotations. An ability to automatically retrieve all annotations to correct something incorrectly identified would be useful.

Note There is a need to consistently go back and revise annotations based on better knowledge and current practice.

Question. (Dirk) Have studies been done of blind comparisons between different annotators?

Ans. (Susan, Kyra). Yes and no. Nothing has been done formally, because annotators do not generally work on the same footage. But informally, annotators discuss among themselves issues and what things may create confusion in order to both share their experience and insure that user needs are met.

Note (Dirk) If we want to compare automated annotation with human annotation, we need to first compare human to human annotation.

Note (General) Movement, behavior, and relative size are keys to identifying critters.

Question (Dirk) Would bit image summary clips of tapes be useful?

Ans. (general) It would be useful to users to see summary samples of what was seen on a dive.

Question (Duane, Dan) In the present system how does one determine which objects are the ones referenced by an annotation?

Ans. (Kyra, Susan) Having to mark something on the screen to associate it to an annotation would add considerable complexity to their task. Right now they use both a mouse and the keyboard to select things or type things to create the annotation using the graphic interface presented by the VIMS/VARS interface. It can be confusing to a user which specific object in the image is being referenced by an annotation when there are several, but there isn't presently an easy way to fix this.

General Comments. There was generally a lot of discussion beyond just the answers recorded here. This discussion was speculative 'what if' kind of things, etc. However several important points were made. Annotators pointed out that they use the audio track of comments made by users as a way of assisting them to determine what is important on a tape. Moreover, pre-annotation is not necessarily much help since annotators generally feel that have to review everything on a tape anyway. Also they are moving towards real time annotation during recording. Tapes from AUV transects or from a cabled based benthic observatory will not have any audio comments or any real time annotation capability. Moreover these tapes are likely to extend over many hours, but with more spacing between interesting frames. Therefore these types of tapes may be a more suitable first target for automated event detection, and annotation assistance.