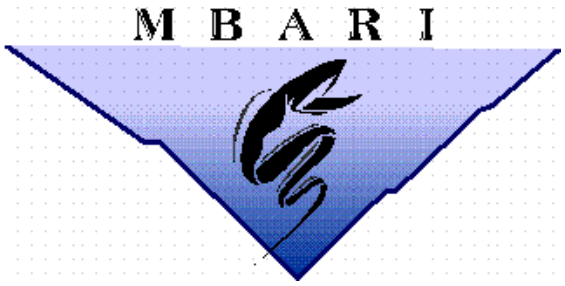
The MBARI Automated Video Event Detection Project



Automated Video Event Detection System

Software Requirements Specification

Version 0.3

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 exchange sections. revised references.

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 list of acronyms that are used below are 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 Video

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. User characteristics

There are two user scenarios: 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 timecodes. The frames collected from this sequence along with detection parameters are then fed into the automated event detection processing system. The event detection processing operates frame-by-frame and outputs event data consisting of images, timecodes, 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 saliency algorithm. 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 then the digitized video must be buffered to enable flexibility in the processing. The output of the automated video event detection 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. In the event display and edit process, it is assumed that the event metadata can be displayed in an overlay of the image. From the event database, clips of the object of interest or a sequence tracking the object over multiple frames can also be generated.

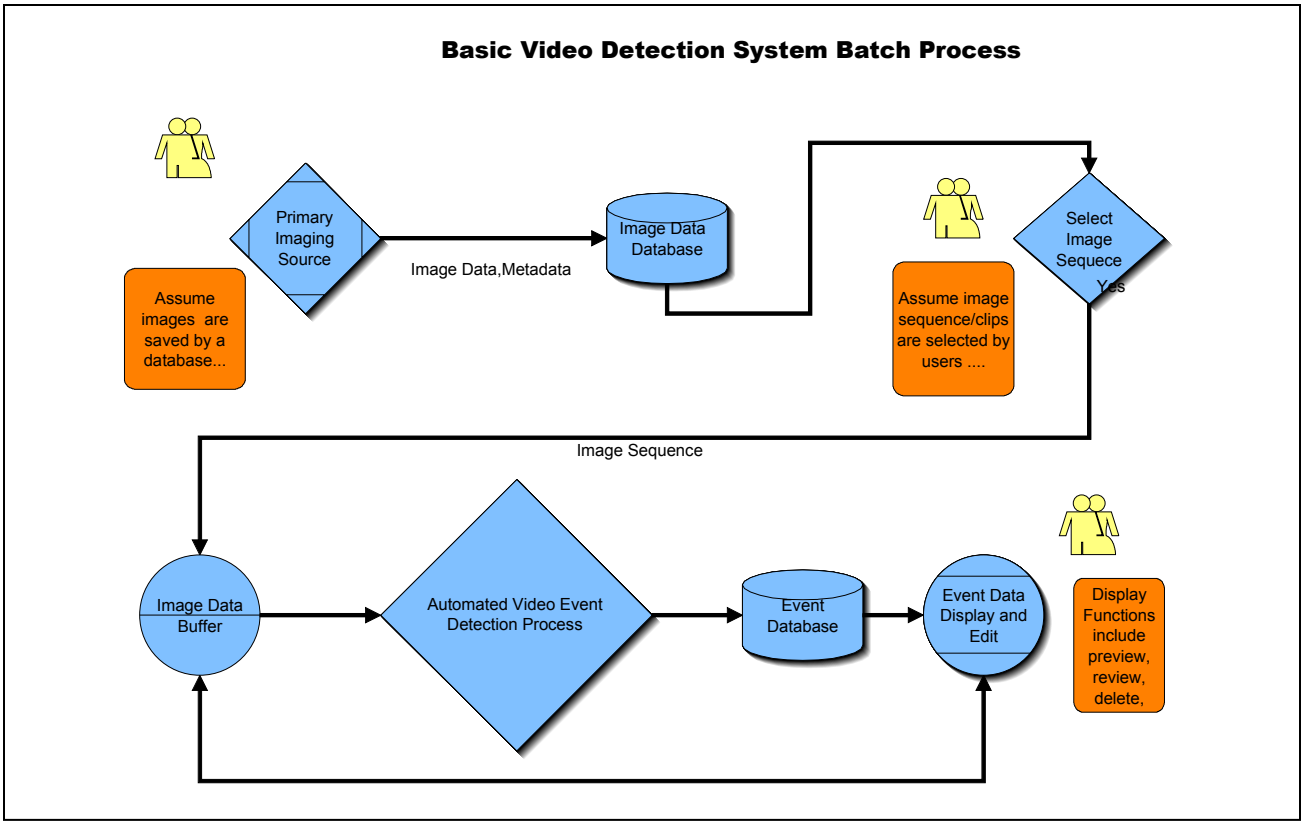


Figure 1. Batch process flow diagram

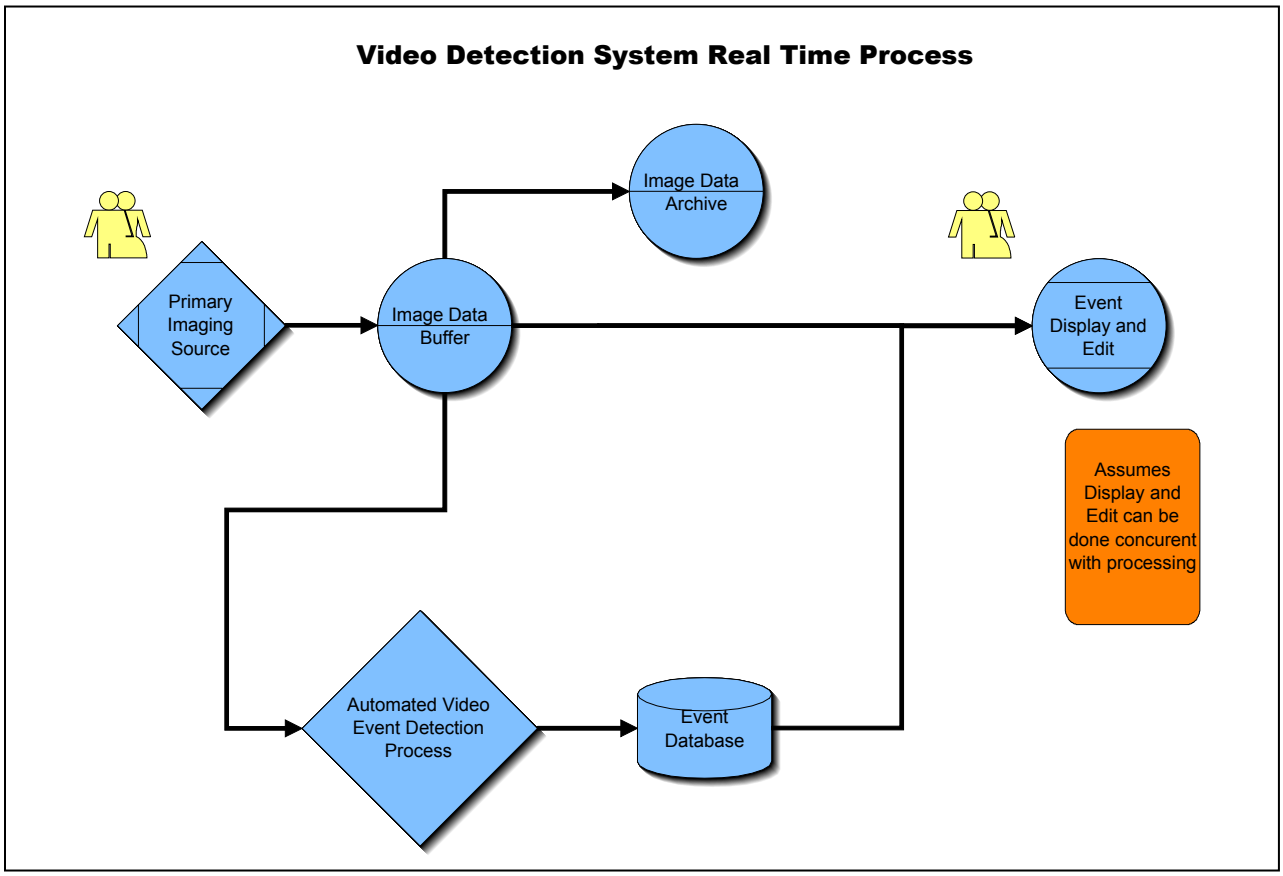


Figure 2. Real time process flow diagram

Figure 2 shows the flow diagram of movement of data through the automated event process in a real-time processing mode. 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 and post-dive batch processed video analyses.

2.3. Constraints

2.3.1. General design

The software is to be designed and implemented in modern, maintainable technology, such as an object oriented programming language (OOP), that 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).

2.3.2. 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.

2.3.3. 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.

2.4. 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 the computing environment required for AVED will be adequate to support the near-real time requirements demanded by some AVED users. It is also assumed 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 capture and buffering must support this frame-based system.

3.1.1. 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 raw RGB colorspace. 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.

3.1.2. 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 type video feed from a tape capture system.

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 source signal. Allowing for a file format like Quicktime, provide a convenient way to move around files. This works well for short clips.

The Quicktime format is an open standard 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 the best 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.

3.1.3. Video buffer capacity

A minimum of ten minutes of data from the primary imaging source shall be stored for processing.

Rationale
Ten minutes is somewhat arbitrary, but based on using AVED as an assistant for analyzing video transect annotations, which are typically ten minutes long.

3.2. Automated event detection and tracking

3.2.1. Visual event and tracking definitions

3.2.1.1. A sequence of objects shall be defined as an event when tracked in more than the user defined *minimum event duration* of frames (see **Data display and storage**)

3.2.1.2. Each event shall be a maximum of 5 minutes and after 5 minutes, if the event is still present, it shall be reset and tracked as new event

3.2.1.3. The AVED software shall detect and track up to twenty visually interesting objects in any given frame in a video sequence

3.2.2. Data storage and display

3.2.2.1. Event metadata listed in **Table 1** shall be stored in XML format

3.2.2.2. The user may specify the XML output file, and if not specified an automatic one will be generated with the UTC day and time.

3.2.2.3. 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 3.4).

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 or Quicktime filename	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 – 180000 frames**		
Minimum event duration	3 – 300 frames	Minimum elapsed time an object must exist before becoming valid event	Y
Minimum event area	10-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 – 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 all frames to remove background objects such as equipment, or other manmade artifacts.	

**** assuming 5 minutes maximum at 60 frame per second input source**

3.3. Exchanging data with other applications

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

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

3.4. Automated video event user interface

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

3.4.1. VCR controls

3.4.1.1. The graphical interface shall provide graphical VCR control to setup the video source for processing

3.4.1.2. 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.

3.4.2. The spreadsheet

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

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

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

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

3.4.3. Editing the event results

3.4.3.1. The user interface shall provide a preview mode. This preview mode will allow the user to seek to successive events and review them, before skipping to the next event. The user should be able to start this preview, by double-clicking on any event, or clicking a preview button

3.4.3.2. The user interface shall provide the user the ability to select and delete an event sequence.

3.4.3.3. The user interface shall provide the user 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.

3.4.4. 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 primary video data source. This may be in the format of an analog or digital video overlay.

3.4.4.1. 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 the display must have good contrast and an adjustable gamma setting, or a default gamma setting that displays natural scenery color well.

3.4.4.2. The overlay display shall display a metadata overlay, consisting of event number, bounding box location, and timecode.

3.4.4.3. The overlay display shall allow the user to toggle on/off the display overlay.

Rationale
The display may interfere with an object display, 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. Automated event classification

Future extensions to the AVED system shall support event recognition and classification. To support future extensions, the AVED processing subsystem shall output event properties, or provide a suitable API to input event properties to event recognition algorithms.

4.2. Estimate the size of a non-moving object

4.2.1. Single camera algorithm
4.2.2. Dual camera algorithm

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 Susna 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.