MVIEWS: Multimodal Tools for the Video Analyst

Adam Cheyer
Artificial Intelligence Center
SRI International
333 Ravenswood Avenue
Menlo Park, CA 94025
+1 415 859 4119
cheyer@ai.sri.com

Luc Julia
STAR Laboratory
SRI International
333 Ravenswood Avenue
Menlo Park, CA 94025
+1 415 859 4269
julia@speech.sri.com

ABSTRACT

Full-motion video has inherent advantages over still imagery for characterizing events and movement. Military and intelligence analysts currently view live video imagery from airborne and ground-based video platforms, but few tools exist for efficient exploitation of the video and its accompanying metadata. In pursuit of this goal, SRI has developed MVIEWS, a system for annotating, indexing, extracting, and disseminating information from video streams for surveillance and intelligence applications. MVIEWS is implemented within the Open Agent Architecture, a distributed multiagent framework that enables rapid integration of component technologies; for MVIEWS, these technologies include pen and voice recognition and interpretation, image processing and object tracking, geo-referenced interactive maps, multimedia databases, and human collaborative tools.

Keywords

Multimodal pen and voice user interfaces, image processing and object tracking, video analysis and annotation, agent architecture.

INTRODUCTION

Although sophisticated tools are now starting to appear that assist an image analyst in manipulating still photos, few systems exist to help an operator efficiently exploit full-motion video. Given video's inherent advantages for characterizing events and movement in a scene, the role of video analysis is taking on increased importance in military, intelligence, and surveillance domains. By considering how video can be best exploited in these contexts, we realize that video analysis poses new challenges and opportunities:

- At the 1996 Atlanta Olympics, after a bomb went off inside of Olympic Park, investigators had to deal with the task of thoroughly searching for clues within some 600 amateur videos related to the incident. Had there been better tools available for indexing, time stamping, annotating, classifying and cross-referencing the videos, this process would have been much more manageable.
- The U.S. military, as part of peacekeeping and intelligence missions, routinely sends unmanned Predator airplanes over target sites of interest. While pilots remotely guide the airplane over the terrain, a second team of analysts is responsible for extracting information of relevance from the video and associated metadata. Although all results of these missions are recorded on videocassette, there is currently no automated method for querying this data repository at a later date. A searchable index would help ensure that this resource is not wasted, and providing the ability to replay audio and written annotations would help the analysts reviewing a video to quickly establish context for what they are seeing.
- In many surveillance or security-related tasks, a single operator is responsible for monitoring the output of many cameras distributed throughout the site. Object detection and tracking, in conjunction with automated alerts and sensor management, can augment the operator's ability to efficiently comprehend and fuse numerous information streams.

In this paper, we present a demonstration system called MVIEWS, for Multimodal Video Imagery Exploitation WorkStation, that attempts to address some of these challenges by bringing together multiple commercial and research technologies into a single toolset. Although each technology is interesting by itself, it is the *integrated use* of these capabilities that can greatly enhance the effectiveness of a video analyst.

SYSTEM DESCRIPTION AND DESIGN

Design Approach

The original design for MVIEWS was based on a vision and on a set of guiding principles. The vision can be described as follows.

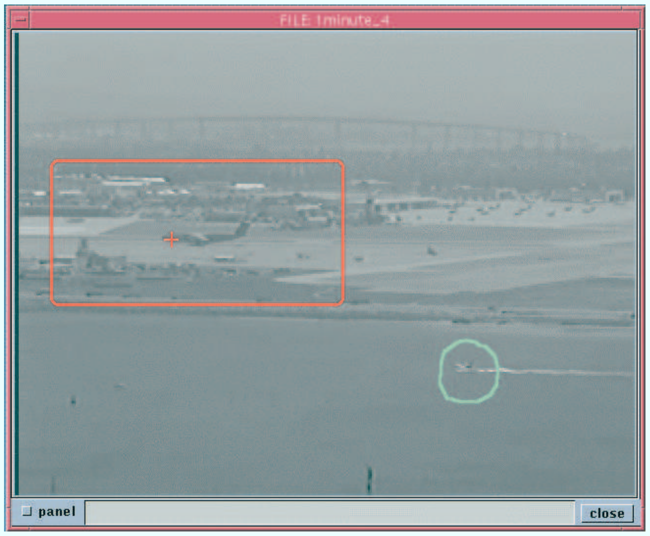


Figure 1. Video Player. Moving objects in surveillance regions are tracked (plane). Multimodal commands can target specific objects (boat).

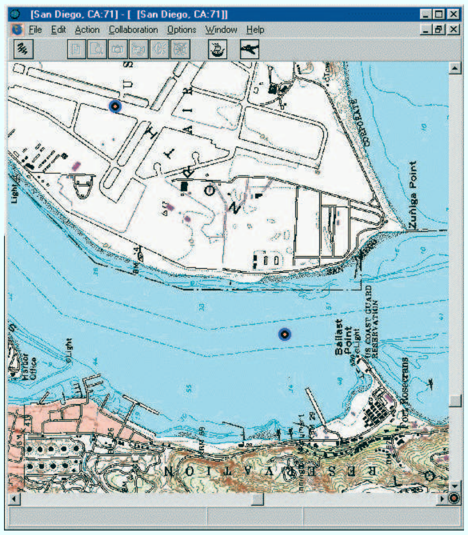


Figure 2. Multimodal Map. Objects tracked in a video are simultaneously displayed on a geo-referenced map.

Imagine a single operator at a terminal, exploiting ten ground and aerial video sensors. Software agents are providing real-time object detection, alerts, queuing, tracking, and information fusion. The operator uses only voice commands and an electronic pen to control the workstation and sensors, to add multimodal annotations to the video streams, and to collaborate with operations and intelligence specialists at remote locations. As a situation develops, agents and humans work together on assessment and characterization, while documenting the process through semiautomatically generated reports.

When setting out in pursuit of this vision, we tried to also keep in mind a few frequently overlooked design criteria:

- People are indispensable. They are good at processing complex visual problems, but they are subject to fatigue and boredom. Automation is an aid, not a replacement.
- User interfaces for complex tasks can quickly become complex. We needed to create as natural, invisible and efficient an interface as possible, combining graphical user interface (GUI) techniques when effective with other, more fluid modalities, such as speech and pen.
- The utility of information greatly increases in conjunction with other supporting information. Thus, we required an open and extensible system that places the needed information and tools at the operator's fingertips.

System Description

The first public demonstration of MVIEWS was given at the Exploitation Technology Symposium (ETS-97) held at Naval Research and Development (NRAD) headquarters in San Diego. Describing this event will provide a good overview of how the current MVIEWS implementation can be used.

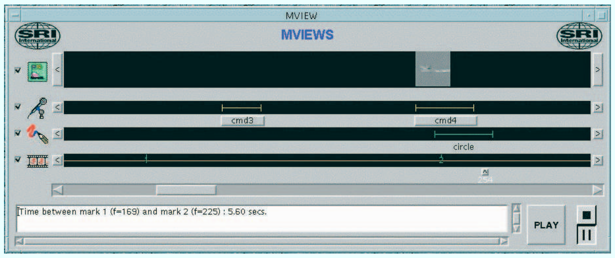
Given the visuals provided by NRAD's location, we chose to exhibit MVIEWS using a border patrol scenario. After securing the necessary permits, we installed a camera on the roof of the demonstration building, such that it overlooked activity in the harbor below and at a nearby military airport. Our video analyst could investigate the movements of small recreational boats, an occasional commercial or military ship, plenty of windsurfers, and a few airplanes and helicopters.

Inside the exposition's demonstration facilities, an operator was seated in front of a Sun[1] Ultrasparc 1 computer, monitoring the live video feed. The operator interacts with MVIEWS by using pen and voice (Figure 1), to perform one of the following functions:

- *Add annotations*, simply by speaking and drawing directly on top of the video. As an example, a surveillance operator might speak «The movements of this motorboat appear unusual» while drawing a circle around the vessel and tracing its path using the pen.
- *Generate reports*, constructed from multimodal input and multimedia data. A typical interaction might contain: «Report, convoy of three vehicles, heading rapidly west along access road.» This information would be saved with the video frame and associated metadata, including time, date, camera type, and spatial coordinates.

---

[1]All product or company names mentioned in this document are the trademarks of their respective holders.



Time between mark 1 (f=169) and mark 2 (f=225) : 5.60 secs.

Figure 3. Media Track Editor. A video is replayed with multimedia overlays, and indexed fields are located in the video or the video database.

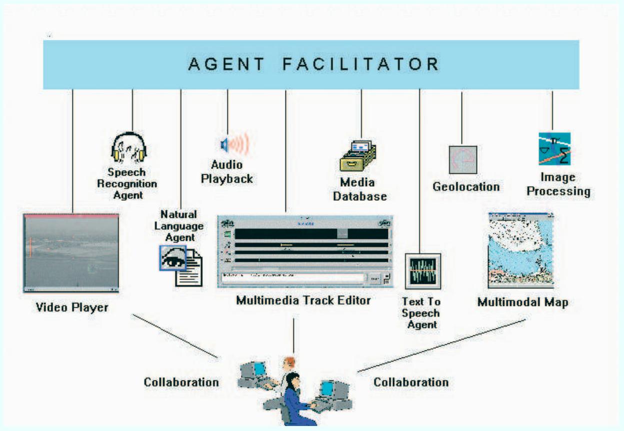


Figure 4. MVIEWS architecture.

- *Specify commands* to the system, involving object tracking (e.g., «Track this boat»), image processing (e.g., «Stabilize image», «Grab this region»), and setting alerts (e.g., «Notify me if this object moves» or «If more than three objects enter this region, alert me»).
- *Collaborate* with remote participants, for example «Bob, can you identify this vehicle type?»

In addition to the video display, the operator could call up an interactive map (Figure 2), simultaneously displaying any objects tracked in the video as geo-referenced points in 2D space. The map display, also controlled through pen and voice, provided additional information about the region. For example, in the Predator UAV domain, the map allows the operator to examine the terrain ahead of the plane's path, and call up supplementary data (e.g., «Show me all military bases near here»).

As a means of attracting further attention to our ETS demonstration, we sent out a person carrying a wireless handheld pen computer to mingle with a reception gathering nearby. After targeting a small group, the demonstrator would show them the map display, look over the balcony and say, «See that boat down there? It's being automatically tracked by software agents from a live video image, and this computer is receiving the reports. Why don't you go in there and check it out?»

Near the first workstation, we positioned another computer where a second analyst could work collaboratively with the first. While one operator monitored and annotated the continuously advancing live video feed, the second was at liberty to provide more detailed analyses of recorded segments of the video. Using the Media Track Editor (Figure 3) as his primary interface, the analyst navigated within the video segment, requested image enhancements, performed timing and distance measurements, and queried the multimedia database for other video segments of interest. The Media Track Editor is structured as a timeline, with annotations, extracted frames and clips, and recognized speech and pen clearly highlighted. By providing an interface for quickly skimming through a video clip and replaying selected sections along with their multimodal annotations, MVIEWS establishes context for what the analyst is examining, accenting what was important to the operator at the time the video was recorded. Instead of fast forwarding through the entire video clip to understand its content, an analyst can save time by perusing annotations from the timeline at different granularities of detail.

IMPLEMENTATION

To implement MVIEWS, we needed to quickly integrate numerous technologies, written in a variety of programming languages, some requiring specialized computer hardware. To facilitate a loosely coupled, dynamic, heterogeneous and distributed integration, we took advantage of the services provided by the Open Agent Architecture™ (OAA™).[2]

The Open Agent Architecture

Similar in objective to distributed object frameworks such as OMG's CORBA or Microsoft's DCOM, a distributed agent architecture such as the OAA can provide integration of components written in different programming languages[3] and running on different platforms.[4] However, OAA agents possess qualities beyond ordinary distributed objects. Agent interactions are more flexible and adaptable than tightly bound IDL[5] method calls in CORBA or DCOM, and are able to take advantage of parallelism and dynamic execution of complex goals. Instead of

[2]For more information about the OAA, see http://www.ai.sri.com/~oaa/

[3] Programming languages: C, C++, Prolog, Lisp, Java, Borland Delphi, and Microsoft Visual Basic.

[4] Platforms: UNIX (SunOs, Solaris, Lynx), Windows (3.1, 95, NT), all Java platforms.

[5] Interface Definition Language: specifies an object's methods using a C++-like header file.

preprogrammed unitary method calls to known object services, an agent can express its requests in terms of a high-level logical description of what it wants done, along with optional constraints specifying how the task should be performed. This information is processed by one or more Facilitator agents, which plan, execute and monitor the coordination of the subtasks required to accomplish the end goal.

The OAA has been used to implement more than twenty different applications, including

- Multi-robot control and coordination [4]
- Office automation and unified messaging [2]
- Collaborative multimodal user interfaces [1, 12]
- Frontends [8] and backends [11] for the Web
- Development tools [10] for creating and assembling new agents with the OAA

Each OAA project can take advantage of the core services provided by the architecture as well as of the growing number of technologies now accessible through an agent interface. These services and technologies include speech recognition, natural language understanding, text extraction, multimodal fusion and reference resolution, reactive planning, virtual reality, image processing, web-related technologies, user modeling, and collaboration tools.

The core services of the OAA are implemented by an agent library, which has been ported to several different programming languages, working closely with a Facilitator agent, responsible for domain-independent coordination and routing of information and goals. These basic services can be classified into three areas: agent communication and cooperation, distributed data services, and trigger management.

Interagent Communication Language

The Interagent Communication Language (ICL) provides the means for interaction among agents. When an agent wants to make a request of the agent community, it describes the goal it wants achieved as well as parameters specifying constraints on how the goal is to be accomplished. The request is sent to a Facilitator agent, which uses the declarative specifications it stores about each agent's capabilities, and the parameters defined for the incoming goal, to produce a fully specified execution plan detailing tasks for distributed agents to perform. The Facilitator agent is then responsible for monitoring and coordinating the execution of the plan, by routing requests (potentially to agents in parallel), collecting results, backtracking when subgoals fail, and finally providing the results to the requesting agent.

ICL requests are expressed using the syntax and semantics of Prolog, a decision influenced by our desire to involve the user as closely as possible in agent interactions. ICL expressions can be generated from the Prolog-based logical forms produced by many natural language parsers, allowing the user to make requests of the agent community in plain English. As a simple example, the English request *«What is the telephone number of John Bear's manager?»* would be converted to the ICL expression:

```
oaa_Solve( (manager('John Bear', M),
            phone_number(M, P)),
            [query(var(P))] ).
```

Parameters can specify both low-level constraints or high-level advice. Examples of low-level constraints might include the maximum amount of time for the solution, the maximum number of solutions returned for a query, how the information should be returned (e.g., as a blocking call or asynchronous streamed response), and rarely, which agents are allowed to participate in the computation.

As an example of high-level advice parameters, notice that the way parallelism should occur depends on the type of task being solved. If three email agents are available on the network, the request «Send this by email to Luc» should probably not be sent to all three agents at once, but rather if the first doesn't succeed, the others should be tried in succession. Compare this with a database query, where it might be very desirable to send the request in parallel to as many available agents as possible. When solving a math problem, different answers returned by different mathematicians could signal a problem, whereas if the participants are students composing poems, varied answers are a requirement.

Data Management

OAA's distributed data facilities share much in common with the distributed goal resolution process described in the previous section. In the same way that agents register the tasks they are capable of performing, agents also declare descriptions of the data they manage. An agent can then add, delete, change, or query a data value, and this request will be automatically routed by the Facilitator agent to the appropriate agent or agents.

Data declarations and functions also make use of the notion of parameter lists. In this case, parameters specify information about permissions, scoping, persistence, whether duplicate values are allowed, and so forth. Data parameters are also used provide synchronous collaboration features to OAA applications; the 'shareable' attribute determines whether a data value is synchronized among all participants of a distributed collaborative session.

Triggers

Triggers allow an agent, or set of agents, to monitor some potentially complex state in the world, performing an action if the trigger's test conditions become true.

Triggers or rules exist in many commercial systems today; for instance, mail programs often allow the user to define actions (e.g., delete, archive, forward) to perform if an email of a certain type arrives. However, in these systems, the action must be predefined and fixed. With OAA triggers, the action part of a trigger may be any compound task executable by the dynamic community of agents. As new, perhaps unanticipated, agents connect to the system at runtime, what the user can say and do literally changes. For instance, if a new fax agent suddenly becomes available, the user can now say or write «If email arrives…, fax it to Bill», even if this action had never been conceived of by the original developers of the application.

Four types of triggers are currently defined by the OAA:

1• Data triggers: *«If the airline flight time changes…»*

2• Time triggers: *«In ten minutes…», «every Thursday from now until Christmas…»*

3• Communication triggers: *«If any agent sends Msg…»*

4• Task triggers (specific to the domain of a particular agent): *«If mail arrives about…», «If this Web page changes …»*

Triggers are stored using the data management facilities, so they can be added, deleted, inspected, protected, and automatically distributed like any other database predicate.

MVIEWS Component Technologies

The MVIEWS application is implemented as a collection of OAA agents, as depicted in Figure 4; we'll now take a closer look at each of the component technologies.

Video Player

Two separate video players have been adapted for use with the MVIEWS system, each with slightly different properties. The first is a public domain UNIX-based program called XAnim, a software-based player capable of displaying numerous video file formats, including MPEG and AVI. The second, MP, was written to take advantage of special libraries provided by Sun to access their hardware video boards. MP can play either from MPEG files or a video source (e.g., live camera or VCR). We are considering integrating yet a third player to provide a Windows PC solution.

The video players were adapted to work within the OAA framework by including the OAA agent library, and by publishing the video players' capabilities using the ICL formalism. In addition, the programs were extended to permit drawing on top of the video by using a mouse or electronic pen.

Speech Recognition

Speech recognition (SR) is used both for entering commands to the system and for extracting content from a user's verbal annotations for video indexing and report generation. The SR technology used by MVIEWS is a large vocabulary, continuous speech, speaker-independent system developed at SRI's STAR Laboratory and then commercialized by a spin-off company, Nuance Communications.[6] The recognizer is based on a hidden Markov model approach, and takes as input a set of models either compiled from a regular expression grammar notation, or constructed through a learning process over a large corpus of data. In the current demonstration system, a grammar defines the set of possible spoken commands, as well as the set of keywords and phrases that can be recognized from annotations or verbal reports.

Natural Language

Two aspects of natural language (NL) processing are used within the MVIEWS system to handle the results from the speech recognition process: NL understanding, to interpret commands to the system, and information extraction from text, to produce indices, summarizations, and reports from spoken annotations.

The OAA is often used to integrate different levels of NL understanding, depending upon the requirements of the system. In most OAA-based systems, prototypes are initially constructed using relatively simple NL components, and as the vocabulary and grammar complexities grow, more powerful technologies can be incrementally added. The current MVIEWS demonstration system has a relatively limited set of commands that are processed by two of our low-end NL systems: Nuance's template-slot tools and DCG-NL, a Prolog-based top-down parser. As the MVIEWS prototype matures, more efficient NL systems can be added, such as Gemini, a robust bottom-up parser based on unification grammars, which interleaves syntax and semantics.

A current DARPA-funded project, which will be folded into the MVIEWS system, focuses on the information extraction task in the Predator domain. Applying SRI's FASTUS [6] and adding better domain coverage for speech recognition will be an improvement over the current implementation, which is based on simple keyword spotting and a hand-coded grammar defining possible reported utterances.

Pen Recognition and Annotation

The pen modality is used in conjunction with speech to add multimodal annotations to a video document, and to issue commands to the system. For commands, a set of pen gestures (Figure 5) can be recognized using algorithms developed in [9].

In our experience, most pen annotations made by users also fall into the class of gestures, usually supplemented by a descriptive audio annotation. Since handwriting has rarely been used, incorporating handwriting recognition into the system has been a low-priority task. However, UNIX-based handwriting recognition libraries have been obtained

[6] http://www.nuancecom.com/

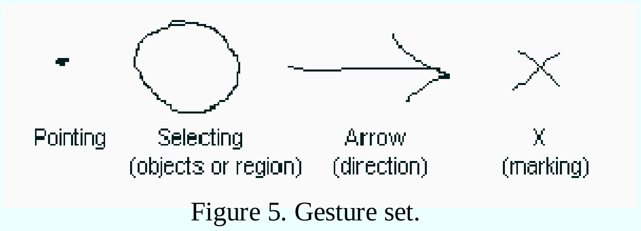


Figure 5. Gesture set.

from Communications Intelligence Corporation (CIC[7]), another SRI spin-off company, and may play an important role for labeling objects with out-of-vocabulary names, a task difficult for speech recognition systems.

Image Processing and Object Tracking

The problem with most image processing and object tracking algorithms is that they are often highly specialized, working well for certain situations and not at all in others. An image and video analyst needs to have an entire library of routines at his or her call.

In the current MVIEWS prototype, we have integrated several image functions, such as stabilization and extraction of selected regions, as well as two object tracking algorithms.

The first of the two tracking algorithms is adapted for detecting fast-moving, relatively small objects within specified surveillance regions in the image. This process requires specialized hardware, running on a Datacube Max Video 200 pipeline image processing system (MV200) with a Motorola 68040 host processor. The Lynx operating system on the MV200 is capable of reading and writing directly in the image memories, using a VME bus. The signal from the incoming video stream is digitized into 256 gray levels and then processed at close to 15 frames per second. Each processing step involves detection of motion between adjacent image frames, followed by temporal correspondence to correlate the moving segments in the video sequence. The strength of the approach is in the temporal association process, which is capable of handling occlusions of moving targets and false alarms from the motion algorithm. As moving targets are detected, their position and ID are passed through the OAA to all interested agents.

The second tracking algorithm, running locally on the Sun Ultrasparc workstation, is good at tracking slow-moving objects, given their initial position (e.g., «Track this car», or «If this boat moves, notify me»). This routine works by comparing via convolution a small subimage of the current video frame with a same-sized subimage from the previous frame. Tracking is initiated by selection of a seed subimage covering the object to be tracked. In our experiments, these subimages ranged in size from 11x11 to 27x27 pixels, depending on the size of the object to be tracked. The object model is formed by storing the location of the subimage within the frame along with the pixel values of the subimage and the square root of the sum of all pixel values within the subimage.

[7] http://www.cic.com/

To find the location of the tracked object in a new frame, we find the local maximum of convolution scores by shifting the model subimage around the neighborhood of its previous location. When a local maximum is found, the pixel values of the subimage centered at that location in the new frame are taken as the new model. The model is updated every frame. The benefit is that the model can adapt to changing views of the tracked object. The drawback is that the algorithm can sometimes be fooled when a tracked object moves into a region where it cannot be distinguished from the background.

Multimodal Map

The multimodal map (MMAP) component of the MVIEWS system allows a user to interact with a dynamic map display through a natural combination of speaking, writing, and drawing directly on the map surface. Multiple modalities may be entered simultaneously or in any sequential order, and merged to produce a command or request. This fusion makes use of the inherent parallelism of the OAA, with multiple agents competing and cooperating to resolve ambiguities arising during the interpretation process.

The multimodal map has been used in various OAA applications, such as providing a natural user interface to travel-related sources on the Web [1], and for guiding and monitoring multiple mobile robots [4]. Adding MMap’s functionality as part of the MVIEWS application demonstrates OAA’s extensibility and flexibility, in that no code had to be written to incorporate the technology into the MVIEWS domain.

Human Collaborative Tools

Within the OAA, a human user will typically interact with distributed software agents, and the agents themselves will communicate and cooperate with each other. A natural extension to this paradigm is to allow multiple human users to work with each other in a collaborative fashion.

The OAA has been extended to include services that facilitate adding synchronous collaborative functionality to any OAA-based user interface. The session management, state replication and an activity-based floor mechanism are provided in a peer-to-peer topology by the Synchronous Collaborative Object Oriented Toolkit (SCOOT) [3], developed by SRI’s Augmented Collaboration Group, or alternatively by a purely OAA-based collaboration mechanism (centralized).

Lessons Learned

Although we have not yet run formal experiments to evaluate the utility of the MVIEWS system, our experiences do suggest several lessons.

The first is simply that that there is a strong need in the Intelligence community for better tools to help an analyst interact more efficiently with video. Video's role in the analysis process is growing: in the case of the Predator UAV, its camera was originally intended strictly as sensor for the pilot to guide the plane, but ended up playing a role in battlefield assessment. In general, the MVIEWS concept has been well received at ETS and elsewhere, and many viewers have suggested domains to which it could be applied.

On the implementation side, two important questions were: how much and what kind of information should flow among distributed agents; and how should we deal with inaccuracies in technologies such as speech recognition or image processing.

Regarding information routing, we chose to limit interagent exchange to messages containing semantic representations of the data, not the data itself (instead of moving video across a network, we split the physical cables so each machine could have a local input source). Agents registered triggers, filters and constraints on broadcast data (as described in the section on OAA) and simple heuristics were inserted for regulating the rate with which information needed to be updated: a fast moving object requires more frequent updates, than a slow one. Clearly, our simple prototype has not solved all hard problems in this area; however the facilitated approach seems promising for managing an efficient data flow.

Regarding recognition technologies, we found that speech and pen are currently reliable enough to be of use for controlling tasks in the user interface, but that for other tasks (speech: transcription of annotations; image processing: object tracking), recognition technologies produce imperfect results given a broad class of input. We chose to design MVIEWS such that as these technologies mature, they can take on an increased role. Multimedia annotation and playback are reliable and useful features by themselves, and provide data on which transcription and classification technolgies can act in the future. An analyst can apply image functions such as stabilization and enhancement with confidence, and then get a sense of when this can be augmented by object tracking or detection. With image processing, a single algorithm will not be sufficient for all input, so we incorporated multiple algorithms, with their use under the direction of the human user. Eventually, we may be able to automate the decision about under which conditions each should be used.

RELATED WORK

Commercially available tools provide much useful functionality for video manipulation and annotation. One good example is Z/Videoware from Z Microsystems.[8] Z/Videoware allows a user to play digital video clips and add audio annotations. It possesses an innovative feature that tries to classify the video by examining the closed-caption text embedded within, and then sorting the video appropriately into different folders.

Another example of a system for video annotation and analysis is VANNA [5]. This application provides a highly tailorable user interface, and an efficient system for annotating the video, using a variety of input devices, such as mouse, trackball, keyboard, touch screen and electronic pen.

Although commercial systems provide some of the functionality that one would want for video annotation, they are missing features that we feel are required for effective video exploitation, such as collaboration, natural user interfaces, and the ability to call up a variety of related data and tools from the same user interface. Although several research systems attempt to apply more advanced technologies to image processing [13], we are not aware of such systems focusing on video.

One effort that has a great deal in common with MVIEWS is a DARPA-funded project called QuickTurn [7], by MITRE and Carnegie Mellon University. MVIEWS and QuickTurn share many of same goals (an integrated environment combining advanced user interfaces with tools and databases for the intelligence analyst) and technologies (collaboration, mediated databases, maps, image tools). However, MVIEWS attempts to focus its solutions within the context of the video analyst (e.g., object tracking, indexed search on pen and voice annotations).

CONCLUSIONS AND FUTURE DIRECTIONS

The current version of MVIEWS can only be considered a working prototype system, but this IR&D project has already shown potential for enhancing the tools and techniques available in the domain of video exploitation and analysis. By using an open, distributed approach as the underlying architecture, we were able to rapidly bring together pertinent technologies, and facilitate the future development of the system as components are added, improved or replaced. Although MVIEWS consists of a number of capabilities that are interesting by themselves, it is the *integrated use* of these capabilities that can greatly enhance the effectiveness of the operator.

Improvements and extensions will be made to the system. One comparatively large project is already under way to improve the speech recognition and data extraction from annotations in the Predator domain, by combining robust speech recognition and the FASTUS text extraction software. Other improvements to MVIEWS will involve a wider use of the collaboration technology, adding more

[8]Z Microsystems' homepage: http://www.zmicro.com/

object tracking and image processing techniques, and identifying and integrating additional video-related technologies. We will also pursue the opportunity to perform user experiments to better quantify benefits of the system.

ACKNOWLEDGMENTS

This IR&D-funded project is the result of contributions from many talented people spanning six centers within SRI. The participants include: Jeff DeCurtins, Gopalan Ravichandran, Bikash Sabata (video and image processing), Greg Myers, Bob Bolles, Eric Rickard, Joel Cain (vision, design and direction), Luc Julia, Adam Cheyer (agent architecture, speech, gesture, handwriting).

REFERENCES

1• Cheyer, A. and Julia, L. Multimodal maps: An agent-based approach. In *Proc. of the International Conference on Cooperative Multimodal Communication (CMC/95)*, Eindhoven, May 1995.

2• Cohen, P., Cheyer, A., Wang, M., and Baeg, S. An open agent architecture. In *AAAI Spring Symposium*, pages 1—8. Stanford University, March 1994.

3• Craighill, E., Fong, M., Skinner, K., Lang, R., and Gruenefeldt, K. SCOOT: An Object-Oriented Toolkit for Multimedia Collaboration. In *Proc. ACM Multimedia '94*, pages 41—49, San Francisco, October 1994.

4• Guzzoni, D., Cheyer, A., Julia, L., and Konolige, K. Many Robots Make Short Work. *AI Magazine*, Vol. 18, Number 1, pages 55—64. Spring 1997.

5• Harrison, B. L. and Baecker, R. M. Designing Video Annotation and Analysis Systems. In *Proc. of the Graphics Interface 92 Conference*, pages 157—166. Vancouver, B.C., May 11-15, 1992.

6• Hobbs, J., Appelt, D., Bear, J., Israel, D., Kameyama, M., Stickel, M., and Tyson, M. «FASTUS: a cascaded finite-state transducer for extracting information from natural-language text,» in *Finite State Devices for Natural Language Processing* (E. Roche and Y. Schabes, eds.), Cambridge MA: MIT Press, 1996.

7• Holland, Roderick. QuickTurn: Advanced Interfaces for the Imagery Analyst. *DARPA/ITO Intelligent Collaboration & Visualization (IC&V) Program PI Meeting*. Dallas, Texas, October 10, 1996. http://www.ito.darpa.mil/Proceedings/icv/agenda.html

8• Julia, L., Cheyer, A., Neumeyer, L., Dowding, J., and Charafeddine, M. HTTP://WWW.SPEECH.SRI.COM/DEMOS/ATIS. In *AAAI Spring Symposium*, pages 72—76. Stanford University, March 1997.

9• Julia, L., and Faure, C. Pattern recognition and beautification for a pen-based interface. In *ICDAR'95*, pages 58—63, Montreal, Canada, 1995.

10• Martin, D., Cheyer, A., and Lee, GL. Agent development tools for the open agent architecture. In *Proc. of the International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology (PAAM).* London, April 1996.

11• Martin, D., Oohama, H., Moran, D., and Cheyer, A. Information brokering in an agent architecture. In *PAAM'97*. London, April 1997.

12• Moran, D., Cheyer, A., Julia, L., and Park, S. Multimodal user interfaces in the Open Agent Architecture. In *Proc. of IUI-97*, pages 61—68. Orlando, Jan. 1997.

13• Srihara, R., Zhang, Z., and Chopra, R. Show & Tell: Using Speech Input for Image Interpretation and Annotation. In *AAAI-97 Spring Symposium, Workshop on Intelligent Integration and Use of Text, Image, Video and Audio Corpora*, pages 17—24. Stanford University, March 1997.