The Electronic Scrapbook:
Towards an Intelligent Home-Video Editing System

by

Amy Susan Bruckman

A.B. Physics
Harvard University
Cambridge, MA
1983

SUBMITTED TO THE MEDIA ARTS AND SCIENCES SECTION, SCHOOL OF ARCHITECTURE AND PLANNING, IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE IN VISUAL STUDIES
AT THE
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
September 1991

©Massachusetts Institute of Technology 1991
All Rights Reserved

Signature of the Author

....
Amy Susan Bruckman
Media Arts and Sciences Section
August 9th, 1991

Certified by

....
Glorianna Davenport
Assistant Professor of Media Technology
Thesis Supervisor

Accepted by

....
Stephen A. Benton
Chairman
Departmental Committee on Graduate Students

The Electronic Scrapbook:
Towards an Intelligent Home-Video Editing System

by

Amy Susan Bruckman

Submitted to the Media Arts and Sciences Section, School of Architecture and Planning,
on August 9th, 1991, in partial fulfillment of the requirements
for the degree of Master of Science in Visual Studies
at the Massachusetts Institute of Technology

Abstract

How many people's home videos remain unedited and unwatched? Home video is a growing cultural phenomenon; however, few consumers have the time, equipment, and skills needed to edit their work. The Electronic Scrapbook is an environment designed to encourage people to use home video as a creative medium. The system and the user collaborate to create home-video stories.

This work addresses issues of knowledge representation and interface design. Semantic knowledge representation is evaluated as a way to represent information about complex, temporal media. A modified form of case-based reasoning, "knowledge-based templates," is used to explore what a computational model of a home-video story might be.

Thesis Supervisor: Glorianna Davenport
Title: Assistant Professor of Media Technology

This work was supported in part by Apple Computer, Inc., and by The Nichidai Fund.

Acknowledgements

I'd like to thank my advisor, Professor Glorianna Davenport, for her guidance and support.

My thesis readers, Professor Kenneth Haase and David S. Backer, Phd. provided thoughtful comments and technical expertise.

Marc Davis, Pattie Maes, Alan Ruttenberg, and Michael Travers commented on drafts of this thesis. It was Marc who first suggested using a "scrapbook" design metaphor. Marc helped to critique the system design as it evolved. Mike and Alan have been a constant source of technical support, answering many a pesky question when I was first learning lisp.

I'd like to thank all of my colleagues at the Media Lab for providing such a wonderful environment to work and play, often the same thing, especially: Edith Ackermann, Carlos Alston, Joe Chung, Stuart Cody, Eddie Elliot, Henry Lieberman, Ron MacNeil, Dave Rosenthal, and Liza Wirtz.

The narrative-intelligence reading group has provided an intellectual context for research at the Media Lab. In addition to those members already named above, I'd also like to thank Professor Henry Jenkins.

I'd like to thank Mom and Bernie for being there for me.

Finally, thanks go to all those who lent me their home videos -- Linda Haviland Conte, Marie Crowley, Chris Gant, friends of Shawn O'Donnell, Rosalind Picard, Chris Schmandt, and especially Dad, Mari, Alicia, Danielle, Rachael, Enid, Rose, Toby, and Lucy.

Contents

List of Figures

List of Tables

1. Introduction

1.1 An Electronic Scrapbook

The Electronic Scrapbook is a computer system designed to help people to use home video as a creative medium. Segments of video can be placed on "pages" of an electronic scrapbook, just as photographs are placed on pages of a paper scrapbook. Segments of video can be played back individually, or whole pages can be played in sequence to form a story. The system helps the user to describe video and arrange it into stories.

There are two sets of motivations for this work, practical and theoretical. On the practical side, there are millions of people who shoot home video but do not have the time, equipment, or skills to edit their work. Once removed from the camcorder, tapes gather dust on shelves and are viewed rarely if at all. Breaking the barriers between consumers and video editing is largely an issue of interface design. Consumers need a system which is easy to understand, fun to use, and designed to help them gradually learn more about the medium.

While The Electronic Scrapbook can be viewed as an editing system, to do so misses an important point: the interactive, electronic format has desirable features. A scrapbook is not merely a means to generate linear videos -- it is a new media form which is an end in itself.

From a theoretical perspective, this work is motivated by issues of knowledge representation. Computers can not yet look into a video signal and identify a tree or analyze an audio track and recognize the sound of wind in the leaves. These are hard problems and their solution is not imminent. Furthermore, even if it were possible to identify a "tree," the computer still needs a way to represent this information. If computers and video are to work together successfully, then computers will need a way to represent information about motion pictures and audio. The Electronic Scrapbook uses semantic knowledge representation to describe a database of home video. This thesis analyzes the benefits and shortcomings of this approach and raises issues for future research.

Lastly, the Electronic Scrapbook explores what a computational model of a home video "story" might be. How can a computer construct a story? How do people construct stories? The Electronic Scrapbook contains some simple "story models" which begin to explore these questions.

The specific subject matter, home video, was selected to highlight these theoretical issues. There are certain archetypical stories which occur in many people's lives -- vacations, weddings, and children are common subjects. It is hypothesized that different people might benefit from using the same set of story models to help them to edit their home videos. In computer terms, this is a knowledge-based approach. The computer can use information about the lives of typical camcorder owners to anticipate user's needs. The regularities inherent in the subject matter make this approach practical.

1.2 A Sample Interaction

Here is a fictional narrative of a stereotypical American family using The Electronic Scrapbook:

It's Monday evening at 9 pm. The week with the kids at the beach was fun, and Susan is just adjusting to being back in her normal routine. Now that Kevin and Tim are asleep, she decides to see how the video of the trip turned out.

She plugs her camcorder into her personal computer and starts up The Electronic Scrapbook. She creates a new page in her scrapbook for the trip and calls it "Our Trip to Asparagus Beach." The first clip she decides is interesting is one of the family piling out of the car at the beach. Kevin, who is seven, is wearing his scuba mask and flippers on the asphalt in the parking lot. She digitizes[1] this video clip and describes it, marking the date, place, people present, and the like. She places this clip on the top of the page.

The next clip that looks interesting is one of Tim learning to swim. Tim, who is five, beams proudly at the camera from inside his dragon-headed safety float. A short, lapping wave startles him, and he grabs frantically at his father Jim's swim trunks. Jim lifts Tim out of the water and grabs the dragon with his other hand. They smile at the camera. Susan marks this segment as the events "learning-to-swim" and "trip-to-the-beach" and clicks on the "guess button." A window of similar segments pops up which includes the segment from the parking lot. Susan picks up that segment and drops it down on the new video segment, which causes the system to copy that description. The date and place are correct, so she doesn't need to enter them again. The only difference is that Kevin does not appear in this segment. She removes Kevin's name from the list of people present and clicks on "OK." Describing the segments she chooses takes less time as she proceeds, because she can use these guessed descriptions.

[1]The system currently uses a laser disc to store and play back video. It is in the process of being converted to a digital-video database like the one described in this section. See chapter 3 for more details.

Susan fills a page with segments from the trip, and she and Jim watch the results. The segment of Tim learning to swim reminds Jim of a segment of Kevin from a couple years earlier. He decides to run the story model "important-firsts." The system creates a new page in the scrapbook and fills it with video segments. It alternates shots of the two brothers, beginning with each boy being born, learning to crawl, eating his first solid food, and the like. At the end it shows a shot of Kevin learning to swim from two summers ago, followed by the shot of Tim and the dragon float from the previous week.

Watching this story, Susan decides she doesn't like the segment of Kevin eating his first solid food -- it's boring. She drags that segment to the trash. When she clicks on the "discards" button, the system pops up a window of segments which were candidates for inclusion in this story, but were not used. She finds one of Kevin eating baby food in which he manages smear it all over his face and throw some at the lens of the camera. This shot is much more typical of Kevin, she decides, and puts it into the story. She saves her changes to the scrapbook and shuts down the computer. Tomorrow evening she'll show the video to the boys and let them make a few new pages in the scrapbook with their favorite shots.

1.3 Overview of This Document

A thesis is, regrettably, constrained to be a linear document. This text is organized in a simple progression which follows the development of the system. The reader may wish to jump from the introductory matter in chapters one and two directly to chapter eight, which describes the story models or "knowledge-based templates." Story models are designed to help the user to construct simple stories.

The first two chapters of this thesis introduce the problems being addressed and describe the system developed. Chapter two also introduces relevant background work in artificial intelligence, learning theory, and visual interface design.

Chapters three through eight describe the system in detail. Chapter three explains the system's basic design and details its hardware and software components. Chapter four narrates how I chose my video source material. The specific material used influenced the system's design. Chapters five contains a detailed description of the software and explains design decisions made.

Chapter six describes in detail the underlying knowledge representation developed for the system. Different lessons were learned in developing each of its areas -- people, people in audio only, locations, events, actions, significant objects, date, camera properties, and subjective properties.

Chapter seven describes how inheritance is used to save the user work in logging footage. This is accomplished through use of a “guess button.” The guess feature is designed to make the system’s behavior easily understandable, making the concept of inheritance accessible to a naive user.

As mentioned above, chapter eight introduces story models . The results of sample story models are compared and evaluated.

Chapter nine places the work in the context of other work done in artificial intelligence. This knowledge-based approach was inspired by research in case-based reasoning and large knowledge-based systems like the Cyc project. Chapter ten introduces a variety of topics for future research. Lastly, chapter eleven draws conclusions about interface design and knowledge representation.

2. Background: Representation and Interface Design

This chapter introduces the initial problem of manipulating video in a computer environment, and presents background information about artificial intelligence, learning theory, and visual interface design.

2.1 Editing Images

Part of the pleasure of home video -- of, in fact, any kind of image making -- is in shooting, the act of creating the image. Another part is in viewing the image and sharing it with others.

For a first viewing, one watches everything that was shot. After vacation, the family may gather together to view the video tapes; at the end of the day, a professional film crew watches the "dailies" [PA 84]; the amateur photographer may flip through a stack of snapshots in the parking lot outside of the photo store.

Subsequent viewings benefit from editing. Not all the film shot is included in the final movie, and not all the snapshots taken are placed in the scrapbook. Home video is not often edited, but would benefit from this process. Reducing the amount of material makes repeat viewings more enjoyable. One can shorten the time required to view the images, improve the average quality by selecting pleasing ones, and create new meanings by juxtaposing them. The editing process is also an end in itself; for example, the act of making a scrapbook is another way to enjoy the snapshots.

2.2 The Search Problem

In order to view a remembered chunk of video, it is necessary to *find* it. Professional editors spend a significant portion of their time describing images so that later they can quickly find particular film clips. The ability to find images easily encourages their creative use.

To find a chunk of video, it is necessary to have a representation of the material available. There are many different kinds of representations. The camera-person's memory of what was shot is a representation. The label placed on the tape (for example, "Italy 1989") is a representation designed to jog the memory of a person -- the camera-person or others with an interest in the event. The camera-person's memory includes context information which may not appear in the video itself -- where and when it was shot, what happened before and after, and the like. Human memory is a powerful representation system, and many other representations are used as aids to human memory.

A description of the content of a video tape is a called a "log." The primary purpose of a log, like any other representation, is to help the user to retrieve the material later. A paper description helps the user to perform the action of retrieving the desired image by telling him/her where to look. A computer description allows the system to find the image, saving the user work.

2.3 Computer Representations

The simplest form of computer representation is to divide video into segments[1] and give each segment a title [Sasnett 86]. Title selection is idiosyncratic. One person's titles may not be meaningful to another, or even to the original person after time has passed. Finding a desired segment or type of segment from a list of titles can be difficult.

Keywords make it easier for a person to find segments with the aid of a computer. This approach is used in *New Orleans in Transition, 1983-1986* [Davenport 87] and in *The Elastic Charles* [BD 89], multimedia documents developed at the MIT Media Lab. In The Elastic Charles, a video segment of a man fishing in the Charles River locks is marked by the keywords "locks," "recreation," and "fish." One could search for other instances of each of these keywords.

This is much more useful than searching through the text of segment titles, but it has limitations. For example, searching for all instances of the keyword "toy" will not return segments with a "stuffed animal" or "bubble stuff." The system does not know that a stuffed animal is a kind of toy. It also does not know that "stuffed animal," "bubble stuff," and "fork" are all objects. The system may use the keyword "parent," but there is no representation for the fact that "parent" is a relationship between two people, or that it has an inverse relationship, "child."

[1]A note on terminology: the word "shot" is part of traditional filmmaking terminology, and is used in two ways. Literally speaking, a shot is the contiguous series of frames from the moment when the camera was turned on to when it was turned off. However, the term "shot" is also used to refer to a set of frames that capture an action.

The word "sequence" also has two meanings. Literally speaking, a sequence is any set of shots combined together. A sequence is also the set of shots which capture an event. Thus, one may refer to the "chase sequence " of a movie.

The term "segment" has come into use in the field of computer manipulation of video. A segment is any set of frames between an in-point and an out-point on a video source. Therefore, a segment could contain a shot or a sequence in all of the senses described above.

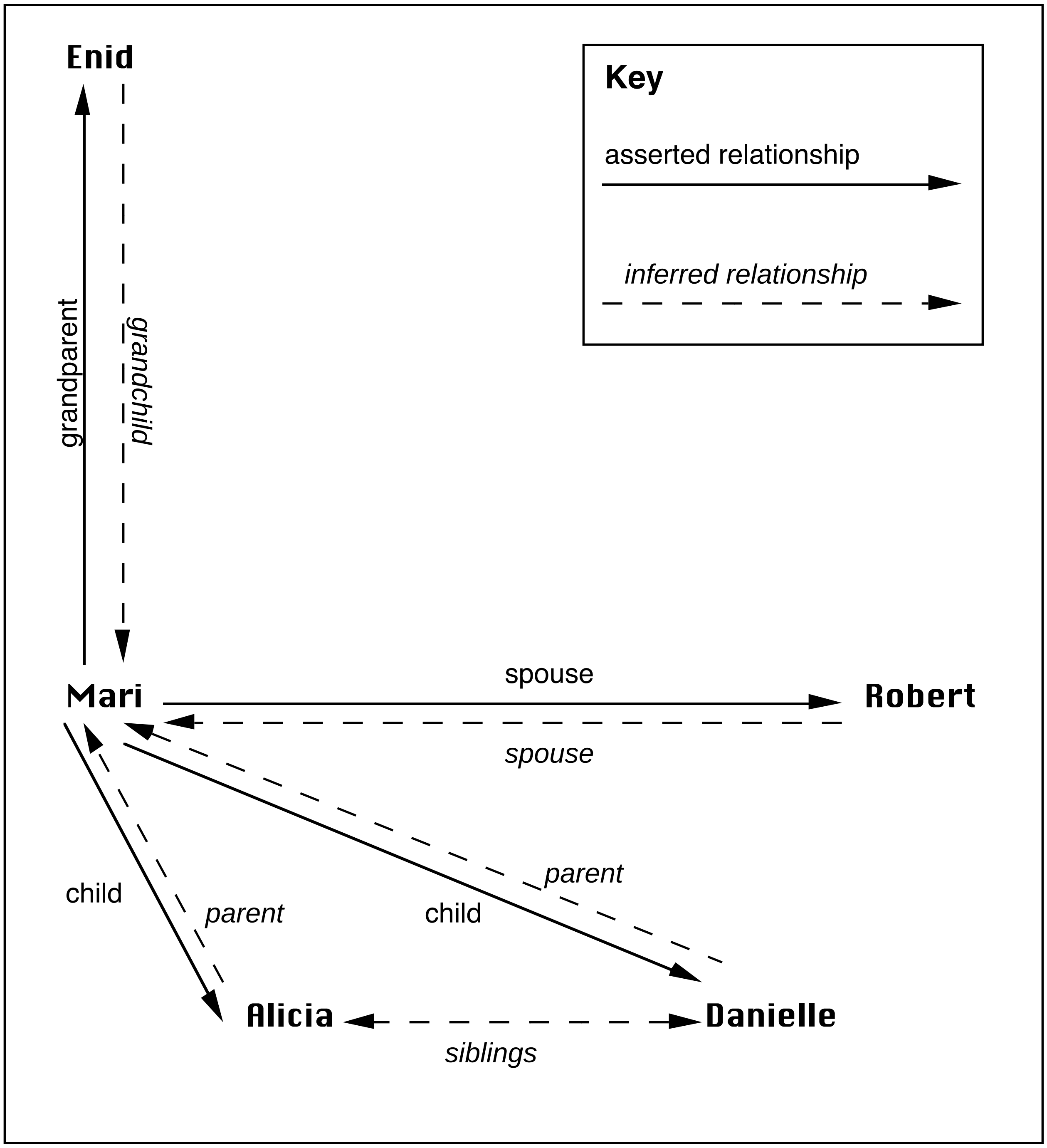


Figure 2.1: A semantic family tree. Mari's relationship to each other person has been asserted by the user (solid lines), and the computer infers the inverse relationships and the relationship between the children (dotted lines). All that is needed to complete the diagram is to assert Robert's relationship to the children. A total of six asserted relationships leads to fourteen inferred relationships.

Semantic knowledge representation[2] can help solve these problems. The Electronic Scrapbook uses **Arlotje**, a knowledge representation system developed by Kenneth Haase [Haase 90]. Searching for all "toys" will return balls, bats, stuffed animals, bubble stuff, and the like, because these objects are defined as subsets of the collection[3] of all toys. Toys in turn are subsets of the collection of all objects. A search for all objects will return forks, knives, and sofas as well as toys.

Properties of units and relationships between units can be tracked. Instead of "Alicia" being simply a keyword, Alicia is an instance of humans with slots that record her birthday and her relationship to other humans.

Inferencing mechanisms help to maintain relationships between units. For example, if we know that Mari is Alicia's mother, then we know that the inverse relationship holds -- Alicia is Mari's child. If it was already known that Danielle is Mari's child, then we know that Alicia and Danielle are either whole or half sisters (see Figure 2.1). Making these inferences saves the user time. Furthermore, family relationships are useful abstractions around which to structure stories.

2.4 How Much Representation is Necessary?

How much representation is necessary? Consider the issue of objects. Is it necessary to represent every object visible in the frame? Is this information useful? A complete description of the objects surrounding me as I write this would fill a small volume: the clutter of books, papers, computer disks, furniture, and photographs. Does it matter that my desk was purchased unfinished four years ago at Market Square Hardware, was painstakingly sanded and polyurethaned by me in the backyard with two coats to the front and four to the top, and that there's a small dent in the back-left corner where it was dropped when I moved? How much detail is useful? The answers to these questions depend on what the representation will be used for. There is no such thing as a completely general representation.

In this particular example, the objects which seem most significant to me are my computer, my desk (extraneous detail omitted), and my cup of tea. These objects play a functional role in the scene. If one were making a home video about me, this would seem to be an appropriate level of detail. This is a subjective judgement.

A representation must be evaluated in terms of its function: does it enable the system to perform the task at hand? What kinds of new tasks might the

[2]A good basic reference on semantic networks is [Winston 84].

[3]For an introduction to the terminology used in Arlotje, see section 3.2 or [Haase 90].

information be used for in the future? Is the representation robust enough to be used beyond its immediate purpose?

2.5 Interface Design and Coding

The primary, original motivation for this thesis was to investigate how to represent knowledge about video. Strategies for knowledge representation, their benefits, and their shortcomings will be discussed throughout this document. As programming work began, however, I discovered that for every 10 minutes of knowledge representation work, there was 10 hours of interface coding.

No information can be used unless it can somehow be entered, stored, retrieved, displayed, and edited. How these tasks can be performed simply and efficiently are not peripheral issues. There is a direct cost/benefit trade-off between the time it takes to do knowledge representation and the tasks the information allows the user to perform. One must also account for the time necessary to learn to use the system; it should not require a computer science degree to edit a video. How can abstract concepts like knowledge representation, electronic signals, and narratives be made understandable to the naive user?

Consequently, this thesis became as much about interface design as about knowledge representation. In this area, the work is inspired by research being done at the Media Laboratory in the Learning and Epistemology Group and the Visible Language Workshop.

2.6 Constructionism

In The Electronic Scrapbook, video is divided into segments which are treated as objects. Seymour Papert's Constructionist theory of learning emphasizes the importance of "objects to think with" [Papert 80]. Treating segments of video as objects is a powerful metaphor. One can make a video segment, put it somewhere, describe it, copy it, or throw it away. These metaphors are clearer to the average user than the concept that one can create a pointer to a section of an electronic signal, put the visual representation of that pointer in a window, create a data structure to be used as an index to retrieve the pointer, make multiple pointers to the same section of video signal, and discard pointers which are no longer needed.

This work is also influenced by the Learning and Epistemology Group's vision of computers as tools to help people learn. It is hoped that through using the system, people will gradually learn about the medium and its potential. The system is designed to promote this process. The typical home video artist does not need a handbook of conventions, but , rather, an environment which encourages him/her to experiment with the medium.

In my experience as a pinball player, I have observed that you can become good at any game that you enjoy losing. In Papert's terms, it is important not to ignore affective properties; learning can and should be fun. I hope that people will enjoy using The Electronic Scrapbook and that it will help them to learn by doing.

In the design of The Electronic Scrapbook, the human and the machine are viewed as partners; each enhances the abilities of the other. Work in artificial intelligence too often tries to create "Einstein in a box" [Hewitt 91] -- systems which do work for people, rather than working with people.

2.7 Visual Interface Design

I began this project with a commitment to making the interface as visual as possible. As the reader will discover, I learned a number of lessons about things that text can communicate more clearly than graphics. A healthy combination of modalities is preferable; each medium should be used for what it does best.

The idea that a medium should be used *only* for what it does best spawned the Modernist movement in art. Post-Modern theory argues that one must try to understand the potential of each medium thoroughly, but feel free to combine them as one wishes.[4] A successful work will communicate with different people on different levels. The difficulty in applying these art-historical ideas to human-computer interface design is that one must work with a more rigorous definition of "communication." In art, communication is subjective; in interface design, it is closer to objective: can the user figure out how to perform the task?

Visual communication is in many ways more challenging than written. Most people spend years learning how to write, but few ever think seriously about visual communication. Certainly, textual communication is preferable to poorly-designed graphical communication. While I do believe that I have developed some insights into the uses of these different modalities, I wish to

[4]The critic Clement Greenberg wrote that each medium should limit itself to what is unique to that medium [Greenberg 61]. Thus, painting should eliminate representation of the real world, because that is the realm of literature. Furthermore, painting should also eliminate representation of the third dimension, because that is the realm of sculpture. Greenberg's writings were influential and helped to spawn movements in abstract art, especially Minimalism. Contemporary critics such as Timothy J. Clarke argue that while a great deal was learned about the properties of various media through these rigid restrictions, the time has now come to discard them and apply the knowledge gained [Clarke 87].

state up front that I am not a trained graphic designer. Research in visual interface design would benefit from greater collaboration between interface designers and graphic designers.

3. A Digital Video Design

This chapter describes the hardware and software used by The Electronic Scrapbook, and explains why a fast-access storage medium such as a digital video database is desirable.

3.1 Current Hardware

The Electronic Scrapbook is being developed on an Apple Macintosh IIx computer with two Macintosh color monitors. The second monitor is being driven by a Mass Microsystems Colorspace II and Colorspace fx board set. These boards allow video to be displayed in a window on the Macintosh monitor.

3.1.1 A Digital Video Scrapbook

Ideally, the system should use a digital video database. The user would hook a camcorder up to the serial port, and use a digitizing board to capture selected segments of video. Putting a video signal in digital form allows it to be stored on a fast-access device such as a hard disk drive. This allows seamless playback of video clips without the gap caused by rewinding a video tape or seeking a laserdisc player. Video can be added or removed from such a system at any time.

Once a video has been digitized, it can easily be viewed, edited, and manipulated. Viewing video on the computer screen could be the final product, or the user could transfer an edited video back to video tape. The edited video could be taken from the digital material; or, alternatively, if two serial-controllable video-tape players are available, then it could perform a tape-to-tape edit from the original source tapes. In video terminology, the digital environment could be used to perform an *off-line* edit, and the system could later perform an *on-line* edit to create a final, high-quality version.

This traditional way of viewing the process misses an important point. The electronic format should not be viewed as merely an intermediate state used as a means to an end. The digital environment has interactive features which are desirable. An electronic scrapbook is a new media form. While the ability to make a video tape is convenient if one wants to send video to Grandma in Oregon, it is an added feature and not the goal of the system.[1]

[1]In the future, it will be possible to send Grandma the electronic, interactive version over a computer network.

3.1.2 Laserdisc as a Stand-in for Digital Video

At the time this project was started there was no reasonable digital video hardware or software for the Macintosh, but it was clear that it would be available in the near future. The Electronic Scrapbook was designed using a Pioneer LD-V4200 laserdisc player as a stand-in for digital video. A laserdisc player has several disadvantages. While it can access video much more quickly than a tape player, it is not fast enough to play clips back seamlessly -- there is a brief seek time in jumping from one part of the disc to another which appears as a moment of black on the screen. Second, it is necessary to send a laserdisc out to be professionally pressed, and once it is pressed its content is fixed. Furthermore, a laserdisc can hold only half an hour of video.

However, the process of editing video down to half an hour to be placed on a laserdisc is analogous to selecting and digitizing video for a digital video database. Laserdisc is an effective temporary substitute for digital video.

In June of 1991, Apple Computer released an alpha version of its QuickTime™ digital video software. Work is currently under way to convert the Electronic Scrapbook to this format.

3.2 Software

The Electronic Scrapbook is being developed in Macintosh Common Lisp (MCL) 2.0 together with **Arlotje**, a knowledge representation language developed by Kenneth Haase [Haase 90]. Arlotje provides a semantic knowledge representation system with sophisticated inferencing capabilities and tracking of dependencies.

The important pieces of terminology in Arlotje are *collections*, *instances*, *slots*, *values*, and *units*. Consider the example of a child's red ball. All toys in the world form the collection "toys." A particular toy, my red ball, is an instance of the collection toys. "My-color" is a slot on objects. The value of the slot my-color on "my red ball" is "red." (Red is an instance of the collection "colors.") Anything defined in Arlotje can be referred to as a unit. For further explanation of this terminology, see [Haase 90].

Arlotje is in many ways more powerful than the Common Lisp Object System (CLOS), the object system which is built into MCL 2.0. Arlotje is used for all significant knowledge representation in this project. However, the MCL window system is written in CLOS. Therefore, all of the interface code for the project had to be written in CLOS. Most interface objects are represented in both systems -- the Arlotje objects have a slot `my-clos-object`, and the CLOS objects have a slot `my-arlotje-object`. This double representation helps to make the two object systems work together in a complementary fashion.

4. Video Source Material

This chapter narrates how I examined a variety of home video, chose my specific source material, edited twelve hours of material down to a two-and-a-half hour rough cut, logged this rough cut using pencil and paper, and edited it further down to half an hour. Lessons learned from going through this traditional process without the aid of a computer helped to shape the design of the computer logging system.

4.1 Choice of Material

In October of 1990 I posted an electronic mail message at the Media Lab asking if anyone was willing to loan me their home videos. I received tapes from eight people which included four weddings, and four tapes of children playing, including two birthday parties. I made copies of thirteen hours of material and took notes as I viewed them.

More people were willing to lend wedding videos, but I turned these offers down; four weddings was more than enough. My first observation from this process was that weddings are boring. The videos varied from twenty-minutes of a traditional Protestant ceremony in the Northeast to four hours of a new-age Jewish ceremony in California with new rituals and vows designed by the participants. Despite this variety, viewing the tapes left me with similar impressions. All of the wedding tapes contained a sea of faces I did not recognize and a set of highly-structured events.

Part of the interest of home video comes from recognizing people one knows. In narrative filmmaking, the information necessary to enjoy a work is usually contained in the film itself; all that is required in addition is some background cultural knowledge. In home video, one also needs to know and care about the specific participants. What appears to be a nondescript elderly couple exchanging pleasantries at the reception takes on new meaning if one knows that Uncle Neil and Aunt Martha have not been on speaking terms since their divorce, or that Neil died a month after the wedding. A viewer who cares about individual participants will also be more tolerant of poor video quality. For these reasons, I chose to use video shot by my family.

Weddings are made up of a sequence of typical events like walking down the aisle, exchanging rings, and kissing the bride. A child's birthday party is another example of an event with a typical structure. This inherent structure can be a guide in editing.

Less-structured information such as children playing provides interesting editing opportunities. A shot of Alicia in the pumpkin patch at age 5 could appear in a video about Alicia growing up, Alicia at age 5, or the family trip to the pumpkin patch. The same segment of video can be used in a variety of ways and takes on different meanings in different contexts. Less-structured information can be manipulated creatively.

It could be argued that these examples of “less structured” information are really just “less likely to be well shot.” A skilled documentary camera-person thinks in terms of sequences -- series of shots that can be used to tell stories. A novice videographer is likely to get enough material to make a coherent sequence of a birthday party -- for example, shots of the candles being put in the cake, the cake being brought out, and the child blowing out the candles can together form a sequence. In the example of a child in the pumpkin patch, the novice videographer is likely to take one shot of the child playing and ignore the approach to the pumpkin patch, the closeup of a pumpkin, and other supporting shots needed to make a sequence. In the absence of the material needed for a proper sequence, the video can be used more abstractly, juxtaposed with video from other times and places.

The Electronic Scrapbook explores ways to structure a video when the raw materials needed to make traditional sequences do not exist. Viewing video of children playing generated a number of ideas. I noticed that a girl named Kaya frequently appeared with two of her favorite objects, her blanket and her stuffed penguin. One could make a video of Kaya growing up by showing shots of Kaya with these objects at different ages. Another way to tell the same story would be to show Kaya at certain important firsts in her life: learning to crawl, eating her first solid food, learning to stand up, and the like. There are different ways to tell the story of a child growing up. One could create a set of story models to help users to discover the variety of stories that can be told.

4.2 My Family’s Video

My half-sister Alicia was born in 1985, and Danielle was born in 1989. My father borrowed a friend’s VHS camcorder to tape Alicia’s birth and subsequently bought a Video 8 camcorder. I borrowed all of their tapes from Alicia’s birth in October of 1985 until Christmas 1990.

Most of the tapes I received from my family were unlabeled, and several had never been viewed after shooting. All together, there was twelve and a half hours of video on ten tapes. A summary of the content of the tapes appears in Table 4A.

Table 4A: General Contents of the Video Source Material

Tape 1:
Alicia being born
Alicia and Mari in the hospital

Tape 2:
New baby shots: nursing, changing diaper, etc.
Christmas 1985

Tape 3:
Christmas 1985, continued
The dogs' birthday
Returning from a hike (Robert and Alicia)
Easter 1986
Rachael and friend making a cake
Enid feeding Alicia
Shots of the old house (San Rafael)
Alicia in the pool
Tour of the new house (Mill Valley)

Tape 4:
Tour of the new house (continued)
Interview with Grandpa about life in Rumania

Tape 5:
Visit to Fire Island
Interviews with Grandpa and Grandma

Tape 6:
Interview with Grandpa
Alicia's first birthday
Everyone's birthday celebration
(Enid's 93rd, Rose's 75th, Robert's 47th, Nancy's 23rd)

Tape 7:
Everyone's birthday celebration (continued)
-- Alicia discovers chocolate
Alicia eating
Alicia playing with Veronica
A visit to the cabin (South Lake Tahoe)
Rachael's junior high school graduation
Amy's college graduation
Crib-club second birthday picnic
Alicia's second birthday (no party)

Tape 8:
A trip to the beach

Tape 9:
Danielle being born
Easter 1989
Danielle at 9 weeks
Danielle at 10 weeks
Alicia playing on the back porch
Mother's day 1989
Alicia's fourth birthday party
(at Chuck-E-Cheez)
Alicia's birthday, continued at home
-- pinata
Alicia and friend playing at being ghosts
Danielle at 13 weeks
Alicia's new nightie
Danielle's first solid food
Alicia and Lonnie in the wading pool
Danielle learning to scoot backwards
Danielle learning to crawl
Danielle and Alicia take a bath together
Danielle's new airplane swing
Enid doing the dishes
Danielle at 6 months, 3 weeks old

Tape 10:
Alicia's first day of school
Alicia's 5th birthday party
(at Chuck-E-Cheez)
Alicia's class singing performance
A visit to a pumpkin patch
Alicia's school gymnastics performance
Alicia and Danielle carving a pumpkin
Alicia's school Halloween parade
Christmas 1990, at the Gilbert's house
Danielle wearing Mari's shoes

There were five primary reasons why the camera was brought out on a given day: a holiday, a birthday, a family outing, a child doing something that she had never done before, or an interview of one of the older members of the family about his or her life before coming to the United States. Video was rarely shot without some special occasion to bring out the camera.

I edited the tapes in chronological order, transferring selected shots to 3/4" tape. I chose shots on the basis of content-based criteria such as importance in family life, and ability to resonate with other shots I had seen, and aesthetic criteria such as lighting and steadiness. The resulting rough cut was two hours and ten minutes long. I made a careful and detailed log of the rough cut. An excerpt from this log appears in Table 4B. The rough cut was later edited down to half an hour, the amount of video that can fit on one side of a laser disc.

I chose to log the rough cut the old-fashioned way, taking notes on paper and then entering those notes into a word processing program. The process of taking notes took approximately eight hours, and typing the notes took seven hours more. The process was grueling and the resulting log contains errors and is not fully detailed.

It was instructive to me to discover how time consuming the process was. While I have logged video for other projects, in the past I had generally made rough notes to jog my own memory. A phrase such as "Alicia and the strawberries" conjures up in my mind an entire sequence which involves some costume jewelry pearls, a sand box, and a slide as well as the watering can and the strawberry plants. While four words are sufficient for my own purposes, a representation that would make sense to another person or to a computer takes up a third of a typed page. (See Table 4B.) Even that would mean little to someone who has not seen the footage and conveys only a fraction of what I remember about it.

Logging camera motion, pans, and zooms was particularly challenging. In professional filmmaking, these factors are usually carefully planned and what was done can therefore be accurately documented.[1] In home video, the camera might jiggle, pan a bit to the right, pan back to the left, and then start to move to follow the subject; the camera properties can be complex and difficult to document. How detailed the documentation should be is an open question.

To describe a segment from the footage of Alicia being born, I copied the entry from the previous segment about that event and modified it. The event, date, people present, source tape, and the like, frequently stayed the same.

[1]Alan Lasky's "Slipstream" project studies continuity supervision, the process of keeping records during professional filmmaking. [Lasky 1990]

While this method saves time, it also leads to errors. For example, objects present in one segment are sometimes incorrectly listed as being in the next segment as well. From this experience, I refined my strategies for how a computer logging system could use defaults to save the user work.

The computer can help to lessen the amount of work required to log video through defaults, semantic inferences, and information derived from audio and video signal processing. Natalio Pincever's master's thesis addresses the issue of obtaining information from the audio signal [Pincever 91].

Table 4B: An Excerpt from the Log of the Rough Cut

This is a short excerpt from the log of the rough cut. The complete log is 35 pages long and describes 2.5 hours of video taken from the original 12.5 hours.

Time:

25:10 Danielle and Baby Mickey

People:	Danielle
In audio only:	Mari
Location:	Home: the kitchen
Objects:	Baby Mickey doll, infant seat
Camera:	Medium shot
Source Tape:	9: Danielle being born
Rough Cut Tape:	2

Mari narrates that Danielle is 2.5 months old.
Danielle plays with a Baby Mickey doll attached to her infant seat.
Mari comments that Danielle looks like the Gerber baby. Danielle smiles.

26:03 Alicia on the porch

People:	Alicia
In audio only:	Mari
Location:	Home: the back porch
Objects:	Watering can, pearls, jungle gym, sandbox, strawberry plants
Camera:	Medium shot; zoom in and out
Source Tape:	9: Danielle being born
Rough Cut Tape:	2

Mari asks Alicia about her new pearls. Alicia denies having pearls on.
Mari asks Alicia to look down at her neck. Alicia says "oh yeah."
Camera pans to Alicia's sandbox and strawberries.
Alicia waters the strawberries.
Alicia climbs on the jungle gym. Zoom in on Alicia's face.
Alicia won't go down the slide because it's wet.

28:19 Curls

Event:	Mother's Day 1989
People:	Alicia, Danielle, Robert
In audio only:	Mari
Location:	Home: Robert & Mari's bedroom
When:	Mother's day 1989
Objects:	A mirror, an infant seat
Source Tape:	9: Danielle being born
Rough Cut Tape:	2

Alicia shows off her hair, which has been curled for the first time.
Danielle is sitting in an infant seat on the floor.
Pan to follow Robert as he takes Alicia into the bathroom to see her hair in the mirror. Pan back to Danielle and zoom in.

5. Interface Design and Development

Writing interface code is time consuming, and maintaining simplicity and consistency are a challenge. The software described here enables the user to control laserdisc players, mark the in and out points of specific segments, annotate segments, organize them on pages of a scrapbook, play back individual segments, and play back lists of segments. This chapter documents in detail the design decisions made, how those decisions evolved from my original conception, and the lessons learned in the process. Issues still to be solved are discussed in chapter 10, Future Work.

5.1 Titled Icons

5.1.1 Combining Text and Graphics

In the original design for The Electronic Scrapbook, logging was to be accomplished primarily by manipulating graphical icons. Thus, to indicate that a particular video segment was taken outdoors, one would drop an icon of a sun onto it (see Figure 5.1).

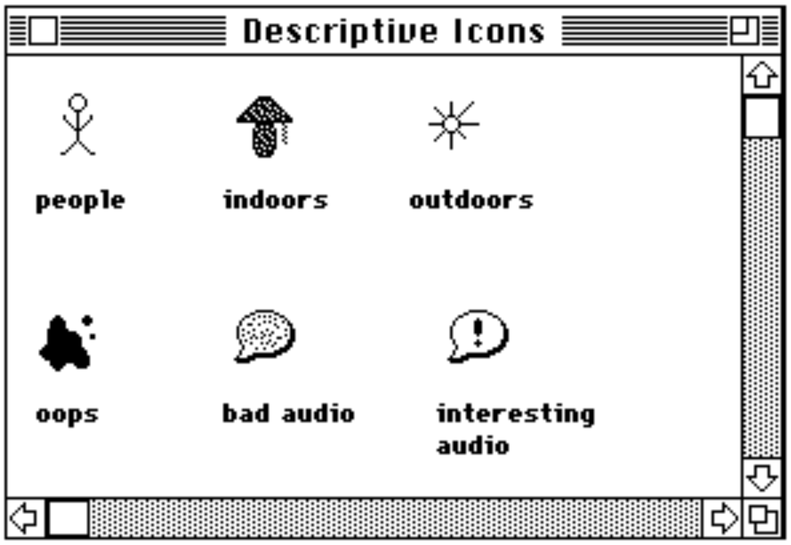


Figure 5.1: The descriptive icons window.

The meaning of icons is not always clear. Users understand the meaning of icons not only from the image itself but also from their expectations and familiarity with similar symbols. It can be difficult to create meaningful icons for unusual or abstract concepts. In his article "The Right Way to Think About Software Design," Ted Nelson writes, "You face a screen littered with cryptic junk: the frying pan, the yo-yo, the bird's nest, the high-button shoe.

Or whatever. You must learn nonobvious aspects of a lot of poorly designed screen furniture and visual toys: what they actually do, rather than what they suggest" [Nelson 90]. To help make the meaning of icons in The Electronic Scrapbook clear, they have been given text titles. Combining modalities can enhance the power of each.

In the process of system development, I discovered that some tasks were better performed with text alone. This issue is discussed in section 5.3.1.

5.1.2 Local Methods

The first part of this programming task was to figure out how to display icons and how to drag them. Next, each icon needed to execute a different action when dropped on a segment. (For example, dropping the "indoors" icon on a segment asserts that the segment took place indoors; dropping the "oops" icon asserts that there is a problem with the quality of the video.) In the Common Lisp Object System (CLOS), methods are defined by object class. In order to make each icon execute a different function when dropped on a segment, it would be necessary to make every icon its own subclass. This approach is awkward if one has a large number of icons or if one wants to generate new icons automatically.

Therefore I chose instead to give each icon a "when-dropped-on-segment" slot which holds a function. This function is executed whenever the icon is dropped on a segment. Separate slots -- when-dropped-on-view, when-dropped-on-segment-editor, when-dropped-on-titled-icon, and when-dropped-on-trash -- determine what is done when an icon is dropped on other objects. Each titled-icon can have a fairly complex, unique behavior. In other words, individual members of a class can have unique *local methods*.

5.2 Video Controller

5.2.1 Device Control

For inspiration in the design of the video controller, I examined the controller Hans Peter Brondmo developed for The Elastic Tools [BD 89] and the Macintosh desk accessory "LD-V4200 Remote," which comes with the Pioneer LD-V4200 laserdisc Player. The resulting controller appears in Figure 5.2.

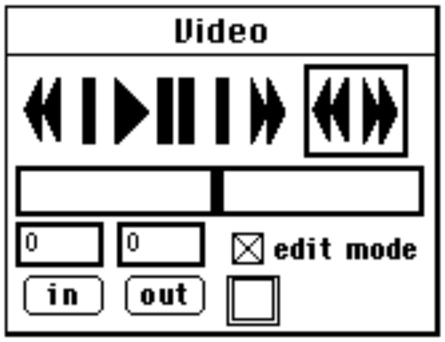


Figure 5.2: The video controller.

The video controller can manipulate multiple devices and device types. A separate controller is created for each device; each controller has a slot "my-device" which indicates the device controlled. The buttons on the controller call generic functions. For example, clicking on the step-forward button calls the function:

```
   (step-forward <my-device>)
```

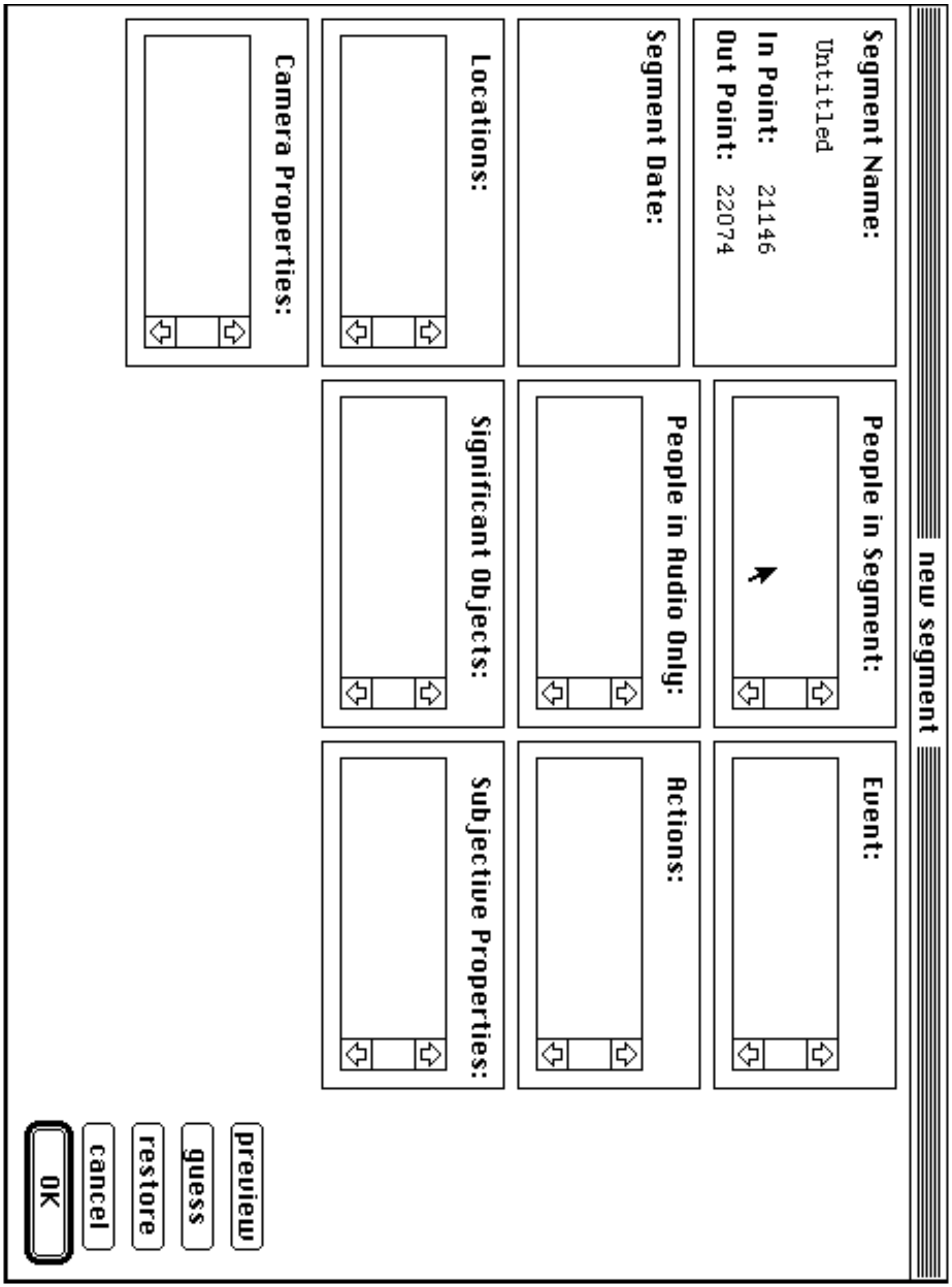
Each type of device has its own step-forward method. To add a new type of device, one merely needs to define a set of methods for how to operate that device.[1]

5.2.2 Paying Full Attention to the Picture

One useful feature of the video controller is the scan bar, which mimics the behavior of the shuttle knob on a standard video edit controller. When the bar is dragged to the right, the player goes forwards at increasingly high speeds, up to the maximum 240 frames/second. Dragging the bar to the left plays increasingly fast backwards. Dropping the bar at any point stops the player. Therefore, one need not look at the controller to locate a specific spot on the video; one can pay full attention to the video while moving the mouse from left to right.[2]

---

[1]Joseph Chung wrote the drivers for the serial port and the Pioneer laserdisc player.
[2]This approach was implemented at the Media Lab by Benjamin Rubin and Daniel Applebaum [Rubin 89]. Their work may be the first example of this interface technique.



Segment Name: Untitled
In Point: 21146
Out Point: 22074

Figure 5.3: The segment-editor window.

5.3 The Segment-Editor Window

5.3.1 Text versus Graphics

In my original conception of the system design, the user would make a video segment by specifying in and out points and clicking on a “make-segment” button. The system would generate a default description of the segment based on what was already known about that video. The user would modify this default description by manipulating graphic icons.

The user needs some way to view the description of a segment. I chose to display an existing description in text rather than graphic form, because text is more compact and less ambiguous than icons. It was a logical step to allow the user to edit this text description directly. This was a slight shift in system design: there would be both text-based and graphical ways to describe a segment.

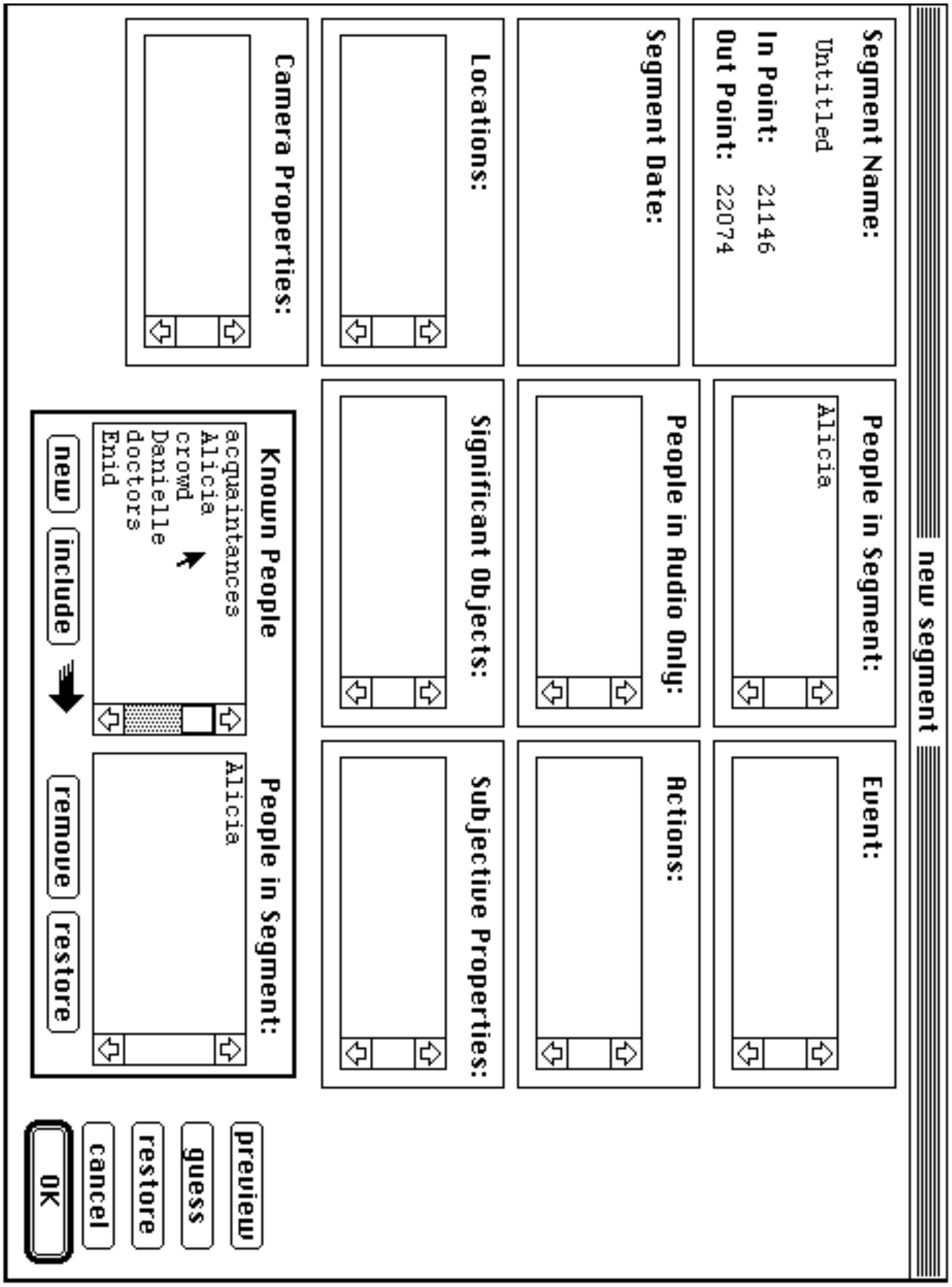
To my surprise, I discovered that the text-based interface was more pleasant to use than the graphical one. A large number of icons are needed to describe video, and it is difficult to organize them so that the correct icon can be found easily.[3] Furthermore, the repetitive action of dragging icons can be tedious. The text-based interface became the primary way to log information.

5.3.2 Panes: Avoiding Screen Clutter

The segment-editor window is divided into ten *panes,* each for a different property (see Figure 5.3). Clicking on a pane puts an editor for that property in the bottom corner of the window (see Figure 5.4).

A pane-based approach avoids the clutter and confusion of having many windows on the screen at once. This design is used on Symbolics computers and may be incorporated into the Apple Macintosh interface guidelines in the future.

[3]Marc Davis’s research on cascading icons addresses this issue. [Davis 91]



new segment

Segment Name: Untitled

In Point: 21146

Out Point: 22074

Segment Date:

Locations:

Camera Properties:

People in Segment: Alicia

People in Audio Only:

Significant Objects:

Event:

Actions:

Subjective Properties:

Known People: acquaintances, Alicia, crowd, Danielle, doctors, Enid

People in Segment: Alicia

Figure 5.4: The segment-editor window, including an editor for the people pane.

5.4 List Browsers

5.4.1 A Sample List Browser

Each pane has its own editor. The editors for all panes (except the date and segment name panes) are specializations of the class list-browser. For example, the following code defines the "People in Segment" pane and its editor:

```
;; People  panes

(defclass people-pane (list-pane)
  ()
  (:default-initargs
    :pane-title "People in Segment:"
    :editor-left-title "Known People"
    :editor-right-title "People in Segment:"
    :view-size #@(170 90)
    :calculate-source-list-function
   #'(lambda (my-list-pane)
       (declare (ignore my-list-pane))
          (get-value *current-session* 'people-in-session))
    :calculate-initial-list-function
   #'(lambda (my-list-pane)
       (with-slots (my-arlotje-object) (view-window my-list-pane)
            (get-value my-arlotje-object 'people-in-segment)))
       :get-existing-value-function
   #'(lambda (my-list-pane)
       (with-slots (my-arlotje-object) view-window my-list-pane)
            (get-value my-arlotje-object 'people-in-segment)))
    :new-function
   #'(lambda (this-button)
       (make-instance
        'person-editor
        :in-video-source
        (slot-value (view-window this-button) 'my-source)
        :my-pane
        (slot-value (view-container this-button) 'parent-window)))
    :my-slot 'people-in-segment))
```

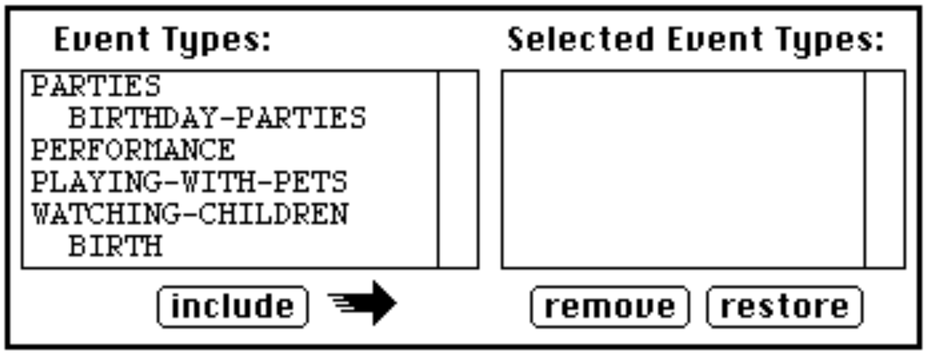
5.4.2 Lists of Options: Towards Logging Consistency

Note that all selection is done from lists; for example, one may either select a known person or add a person to the system. This helps to insure that things are referred to consistently; for example, it is less likely that the user will create separate units for "Alicia" and "Ali" as if they are two separate people.

5.4.3 Hierarchical Sorting: Visualizing the Underlying Representation

A birthday party is a kind of party. The event "birthday-party" inherits properties from the event "party." For example, all of the "likely-scenes" for a party -- guests arriving, eating, dancing, guests leaving, and the like -- are also likely for a birthday party. Birthday parties also have additional scenes -- blowing out the candles on the cake, singing happy birthday, etc. Inheritance is a useful and efficient technique. For more discussion of issues of inheritance, see chapter 7.

The average home video artist will not be familiar with these concepts, but it is possible to make them understood. Lists in list-browsers can be sorted hierarchically. "Birthday-party" appears indented under "party" (see Figure 5.5). "Too-dark" appears indented under "problem-with-video."



```
PARTIES
  BIRTHDAY-PARTIES
PERFORMANCE
PLAYING-WITH-PETS
WATCHING-CHILDREN
  BIRTH
```

Figure 5.5: A hierarchically-sorted list-browser.

When an event is selected or double-clicked, the likely scenes for that event are listed under it. They are displayed in a smaller font, indented under the event, and sorted into their typical chronological order. For example, the alphabetically sorted list of events at a birthday party is confusing:

```
BLOWING-OUT-THE-CANDLES
CAKE-ARRIVING
CUTTING-THE-CAKE
DANCING-SCENE
EATING-SCENE
EATING-THE-CAKE
GIVING-PRESENTS
GUESTS-ARRIVING
GUESTS-LEAVING
OPENING-PRESENTS
LIGHTING-THE-CANDLES
PLAYING-GAMES
PUTTING-CANDLES-IN-THE-CAKE
SINGING-HAPPY-BIRTHDAY
THE-CAKE-SCENE
THE-MESS-AFTERWARDS
WATCHING-PERFORMER
```

However, the meaning of the hierarchically/chronologically sorted list is clearer:[4]

```
GUESTS-ARRIVING
DANCING-SCENE
EATING-SCENE
GIVING-PRESENTS
OPENING-PRESENTS
PLAYING-GAMES
THE-CAKE-SCENE
  PUTTING-CANDLES-IN-THE-CAKE
  LIGHTING-THE-CANDLES
  CAKE-ARRIVING
  SINGING-HAPPY-BIRTHDAY
  BLOWING-OUT-THE-CANDLES
  CUTTING-THE-CAKE
  EATING-THE-CAKE
WATCHING-PERFORMER
GUESTS-LEAVING
THE-MESS-AFTERWARDS
```

5.5 Video Segments

Once created, video segments are placed in the *segments window* (see Figure 5.6). The icon for a video segment is an empty picture frame. In the future, segments will be represented by miniature digitized frames of video. This will allow segments to be recognized more easily, enhance the visual appearance of the system, and make the look and feel of the system more like a traditional scrapbook.

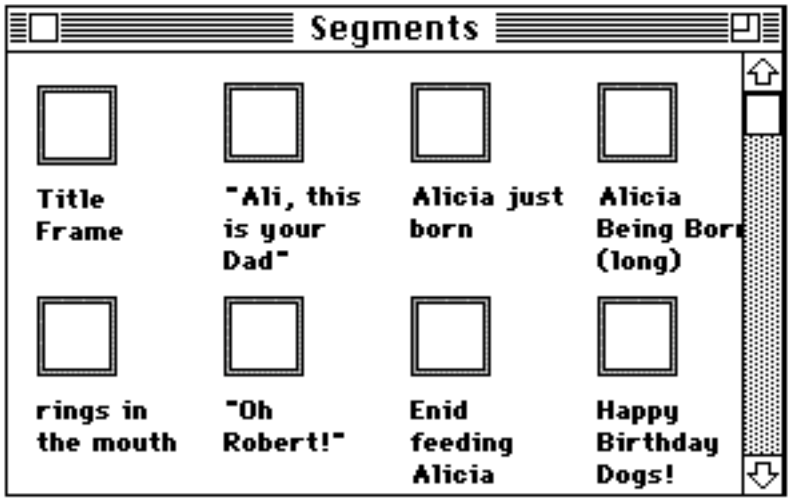


Figure 5.6: The segments window.

[4]Scenes with no usual order relative to eachother are sorted alphabetically.

Clicking on a video segment allows the user to drag it within the segments window, to a scrapbook page, to the trash, or the like. When the shift key is depressed, clicking on a segment plays back that video.

5.6 Scrapbook Windows

5.6.1 Spatial Implies Temporal Organization

Segments of video can be placed on pages in a "scrapbook window" (see Figure 5.7). Clicking on the "play" button replays all the video on the page in sequence -- left to right across rows, starting with the top row and moving down. In other words, the two-dimensional spatial organization of segments on the scrapbook page determines the one-dimensional temporal order of the video.

A scrapbook is made up of a set of pages forming a linked list, analogous to a stack of cards in Hypercard. In the future, it will be possible to generate text headings both manually and automatically.[5] This feature is not implemented yet.

[5]Each story model will have a method for generating an appropriate title for the page. See section 10.6.

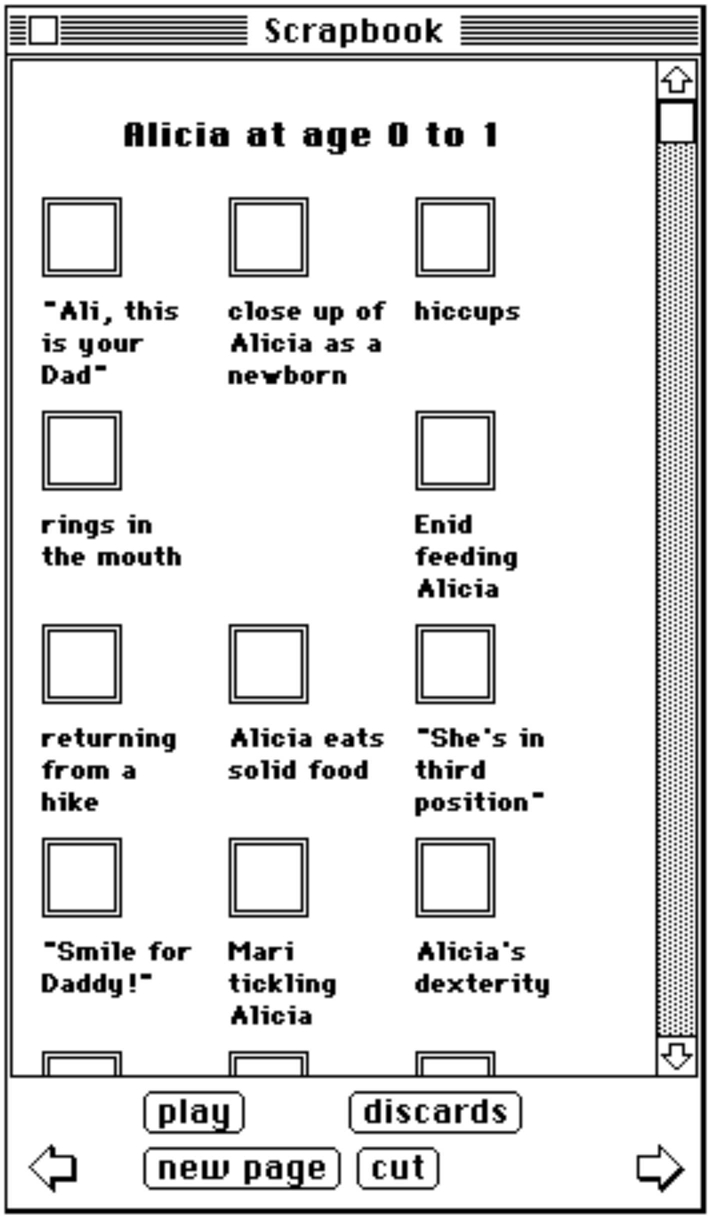


Figure 5.7: A scrapbook window.

6. A Suburban Ontology

Knowledge representation is slave to the interface and servant to the task. No knowledge can be used if it can not be entered and displayed. Furthermore, knowledge is not necessary unless it can be used to accomplish a task.

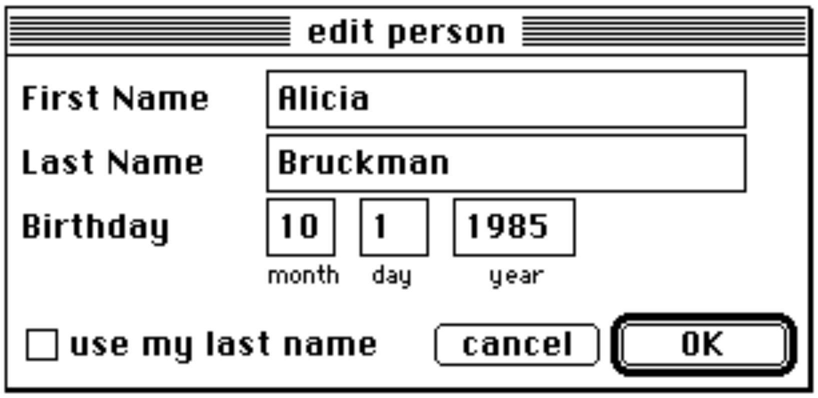
After the video source material was viewed and edited, a draft ontology was developed. This ontology was refined as it was used to log video. Its design is strongly influenced by the specific video source material being used. This chapter will document the current state of the representation, issues that arise, and suggestions for future development.

The base ontology is divided into ten categories: basic segment data (including the segment name, in and out points), people, people in audio only, locations, event, actions, significant objects, date, camera properties, and subjective properties. This base representation can be extended by the user. For example, the system provides a location category "vacation-spot." The user may add "The-Grand-Canyon" as an instance of vacation spot. Spatial representation will be discussed further in section 6.3.

6.1 People

6.1.1 Extending the Representation: New Buttons and Object Editors

The system begins with an empty list of people. The user creates people by clicking on the "new" button. This creates a "new person" dialog box in which the user may enter the person's first name, last name, and birth date. Double-clicking on the icon for a person recreates the same dialog box, enabling the user to edit the values (see Figure 6.1).



edit person

First Name: Alicia

Last Name: Bruckman

Birthday: 10 (month) 1 (day) 1985 (year)

Figure 6.1: The "edit person" dialog box.

Note that a person's first name and last name are values on slots and not the name of the unit itself. This makes it easier to change a person's name. It also facilitates managing when to use a person's full name or just their first name. If the box on the person-editor "use my last name" is checked, then the person will always be referred to by his/her full name. This mimics how duplicate first names are often handled in social situations. A family member named Bernie is referred to as simply "Bernie," and a family friend named Bernie is referred to as "Bernie Zucker."[1]

As was discussed in section 5.4.2, maintaining a list of known people helps to assure consistency of logging; it is less likely that the same person will be referred to by two different nicknames. It also allows the system to track information about people, such as their birthday and family relations.

6.1.2 Inferencing Capability: Family Relationships

Family relationships are a useful data abstraction for home-video stories. One might want to tell a story about a group of siblings, a nuclear family, an extended family, or the like. In some cultures, cousin, uncle, or aunt relationships have great importance, and those relations might be used to help construct stories.

A benefit of using a semantic network is the ability to make inferences. For example, if Mari is Alicia's parent, then the inverse relationship holds: Alicia is Mari's child. Furthermore, if it was already known that Danielle is Mari's child, then it follows that Danielle and Alicia are either half or full sisters (see Figure 2.1).

The computer's ability to make these inferences saves the user work, and also makes the system seem "smarter." Researchers working on the Cyc Project are trying to determine whether, given enough information and inferencing capability, something greater might emerge from a system, perhaps something resembling "common sense" [LG 90]. When a semantic network is implemented on a small scale, however, what is gained is efficiency -- the relationships inferred by the system could be entered by a human. This extra efficiency may be essential to usability of a system. However, it is useful to remember that there is nothing magic about inferencing capability; it merely saves the user work.

[1] It would be possible for the system to check lists of people for duplicate first names in first-name-only mode, and automatically add last names to avoid ambiguity. This feature is not implemented yet.

6.1.3 The Need for a Graphical Interface

Family relationships are currently logged in lisp code, because a satisfactory interface has not yet been developed. To state that Mari is Alicia's mother, one can type:

```
   (assertion (human-name-to-unit "Alicia")
              'mother
              (human-name-to-unit "Mari"))
```

What would interface code for this property look like? One simple solution would be to add fields for a few key family relations to the person-editor object. Thus, there could be "mother," "father," and "children" fields with either pull-down menus or list-browsers to select from lists of people to fill those slots. There are two problems with this solution. First, it limits the number of relationships which can be asserted. Thus, to say that Enid is Alicia's great-grandmother, it is necessary to create the intermediate family members and assert the information indirectly.

More importantly, there is no natural way in this scheme to inform the user of the inferences being made. If the system is now told that Mari is Danielle's mother, it will infer that Danielle and Alicia are sisters. However, the user will not be notified that this inference has been made. There is no way for the user to visualize the underlying family tree and watch it evolve. The user will be unaware of gaps or errors in the representation.

A graphical representation would help to solve these problems. People can be represented as points or icons on a plane. To assert a relationship, the user selects the relationship and drags an arrow between the two people. Inferred relationships are then drawn in a different color or line thickness.

Graphical methods are a common way to display and manipulate information in a semantic network. One problem commonly arises: if all relationships are marked, the display of even a simple semantic network can become so crammed with information as to be illegible [Travers 89].

This problem can be eased by giving the system a representation of which relationships are relevant to display and hiding those that are not currently needed. For example, the default might be to display parent-child relationships but not grandparent-grandchild relationships. The user then can change which relationships are displayed in order to accomplish a specific task.

6.1.4 But What Can All of This Information Be Used For?

In the abstract, it seems useful to be able to track family relationships in a home video system. However, for the quantity of video information the Electronic Scrapbook currently uses, this information is useful but not essential. Nevertheless, I believe that the benefits increase as the amount of available data increases. If many people are present in the video database, then family relationships may be a powerful abstraction. This hypothesis needs to be tested with a larger volume of data.

Consider this example. Knowing family relationships, it is possible to make a story model for "child-with-siblings" which focuses attention on the child selected and includes a roughly equal number of shots of the child with each of his/her siblings. This is equivalent to a more general model, person-with-others; the only difference is that the system is calculating the list of others. The representation of siblings has two benefits: it suggests a possible way to tell the story to the user, and it saves typing time if there are a large number of children in the family.

The system currently has a representation for a person's immediate family and extended family. Your immediate family is defined as your parents, siblings, spouse, and children. Your extended family is defined as the kleene-star of immediate family; in other words, it recursively includes the immediate family of your immediate family.

In a video with a large number of people present, this information could be used to help focus attention on the important people. In fact, the system has a representation of who the important people are at different types of events. For example, the important people at a wedding are defined as the members of the wedding party plus the immediate families of the bride and groom. The important people at a birthday party are the immediate family of the guest of honor.

6.1.5 Focus Person

It would be useful in the future to add the ability to mark the focus person for a video segment. Consider, for example, a shot of a child learning to walk. The child's parents and older siblings are present in the scene, and everyone is walking. Merely checking who is performing the action "walking" will not reveal the key information: this is a shot of the youngest child learning to walk. The solution is to allow the user to mark some of the people in a shot as more important.

For the example of a child learning to walk, the Electronic Scrapbook currently finds the important person by checking ages of the participants. If there are a one-year-old, five-year-old, and two forty-year old people in a shot of the event "learning to walk,"[2] it is likely that the one-year-old is the focus person. This assumption could, of course, be wrong: for example, one of the adults could be recovering from a stroke. However, even without morbid counter examples, the focus person is an important and useful concept and it is preferable to represent this information explicitly. The inferred value should be made a default which can be changed by the user.

6.2 People in Audio Only

6.2.1 The Need for Separate Representations of Video and Audio

It is often the case that someone speaks but does not appear in the video. For example, the person holding the camera may narrate or engage in conversation. If Robert is narrating, he is present but one would not want the shot to turn up in a search for video of Robert. He can be logged as being present in audio only.

This phenomenon is only the most obvious manifestation of a more general problem: the audio and video can have different content. Consider, for example, this video segment: Mari and Alicia are in the pool. Alicia is splashing in the water. Robert narrates "Ali can crawl backwards, but not forwards yet. She climbed one step today in the new house." What is happening in the picture is different from what is being discussed. Ideally, the content of the audio needs to be represented separately from the video.

6.3 Locations

6.3.1 Functional rather than Geographic Information

Locations for home video center around the home, local recreational facilities, and vacation spots. In home video, the functional properties of a place tend to be more important than the geographic ones. In other words, it is more useful to know that the Grand Canyon is a tourist attraction than to know that it is in Northwestern Arizona.[3] The basic locations hierarchy now is:

```
COMMERCIAL SPACE
  HOTEL
  OFFICE
  RESTAURANT
```

[2]See section 6.4 for further discussion of events.

[3]If specific geographical information was needed, then a map-based graphical interface would be useful. A tool for anthropological or geological research, a news archive, and a large stock footage archive are examples of applications in which this information might be useful.

```
    STORE
  INDOORS
  INSTITUTIONAL SPACE
    GOVERNMENT-OFFICE
    HOSPITAL
    PLACE-OF-WORSHIP
    SCHOOL
  OUTDOORS
  RECREATIONAL-SPACE
    BEACH
    LAKE
    MOUNTAINS
    PARK
    POOL
    RIVER
    STADIUM
    TOURIST-ATTRACTION
    VACATION-SPOT
  RESIDENTIAL-SPACE
    FRIEND-OR-RELATIVES-HOME
    HOME
    VACATION-HOME
  ROOMS[4]
    BALCONY
    BATHROOM
    BEDROOM
    KITCHEN
    LIVING-ROOM
    PORCH
    YARD
  STREET
```

By using the “new” button, a user can create subclasses of locations to represent specific places. If Italy is defined as a vacation spot, Rome as a part of Italy, and the Spanish Steps as a part of Rome, then a search for the location “Italy” will include video shot on the Spanish Steps.[5] However, all user-created classes are treated by the system in the same way as the parent class. In other words, “the-grand-canyon” and “the-spanish-steps” are both simply specific instances of tourist-attraction.

6.3.2 Linking Space and Time

It would be interesting in the future to link the representations of space and time. All video of Italy must be from August 1987, the month that my family took a trip there. Video of San Rafael can be assumed to be from before 1986, the year that they moved to Mill Valley. In a personal context, space and time are linked.

[4]It would be preferable if rooms appeared hierarchically below other locations in the way that scenes appear below events. (See section 5.4.3.) This change will be made in the future.
[5]The new button for locations has not been implemented yet; new locations must be hand coded. Only the new button for people has been completed.

6.3.3 Interface Coding for a Multi-level Representation

The representation of locations does not yet include slots for what is a part of what. For example, to say that the video of Danielle eating her first solid food was shot at home in the kitchen, one would now mark "home, kitchen, indoors." A preferable representation would be to have the kitchen as a part of home. Furthermore, indoors versus outdoors should be a slot on locations with a default value. Kitchens are assumed to be indoors and balconies are assumed to be outdoors, unless otherwise indicated.

The knowledge representation required to implement these features is easy; the interface coding is a challenge. How is the user to be informed that the system is assuming that a balcony is outdoors? There is, in fact, an indoor balcony in my family's home videos. One solution would be to automatically annotate selected locations with assumed properties. Assumptions could be displayed in a different font or color to attract the user's attention. This approach will work only if there is a reasonably small number of inferences being made.

6.3.4 Multiple Inheritance

Although indented lists can be used to represent single inheritance, it is difficult to use this form to represent multiple inheritance. For example, if I shot video at my friend's cabin in Vermont, I might want to create a location friends-vacation-home which inherits from both friend-or-relatives-home and vacation-home. A graphical representation could display this information more naturally than a list. In the current system, multiple inheritance is accomplished by simply marking all appropriate locations in a list.

6.4 Events

In use of The Electronic Scrapbook to date, the events category has proved to be a powerful and useful abstraction. Events are similar to Roger Schank's work on scripts [SA 77]. Like scripts, events are made up of scenes. Double-clicking on an event adds a list of its likely scenes below that event, as was discussed in section 5.4.3. The scene list for one event type, birthday-parties, is included below.

"Trip-to-a-pumpkin-patch" is an example of an event that would be added by the user. Since the user-added category "trip-to-a-pumpkin-patch" inherits from "family-outings," it has the same scenes as family-outings.

The events hierarchy is:

```
DAILY-ROUTINES
  COOKING-SCENE
  TAKING-A-BATH
FAMILY-OUTINGS
  DAY-AT-THE-BEACH
  PICNIC
  TRIP-TO-A-PUMPKIN-PATCH
  TRIP-TO-THE-ZOO
FILMING-SCENERY
HOLIDAY
  CHRISTMAS
  EASTER
  FATHERS-DAY
  HALLOWEEN
  JULY-FOURTH
  MOTHERS-DAY
PARTIES
  BIRTHDAY-PARTIES
    GUESTS-ARRIVING
    DANCING-SCENE
    EATING-SCENE
    GIVING-PRESENTS
    OPENING-PRESENTS
    PLAYING-GAMES
    THE-CAKE-SCENE
      PUTTING-CANDLES-IN-THE-CAKE
      LIGHTING-THE-CANDLES
      CAKE-ARRIVING
      SINGING-HAPPY-BIRTHDAY
      BLOWING-OUT-THE-CANDLES
      CUTTING-THE-CAKE
      EATING-THE-CAKE
    WATCHING-PERFORMER
    GUESTS-LEAVING
    THE-MESS-AFTERWARDS
PERFORMANCE
PLAYING-WITH-PETS
WATCHING-CHILDREN
  BIRTH
  BOTTLE-FEEDING-BABY
  CHILD-PLAYING
    CHILD-PLAYING-WITH-FRIEND
    CHILD-PLAYING-WITH-RELATIVE
    SIBLINGS-PLAYING
  FIRST-SOLID-FOOD
  HOLDING-BABY
    BURPING-BABY
  LEARNING-TO-CRAWL
    CRAWLING-BACKWARDS
  LEARNING-TO-SIT-UP
  LEARNING-TO-STAND-UP
  LEARNING-TO-SWIM
```

```
    LEARNING-TO-WALK
      FIRST-STEPS
    NEW-FOOD
    NURSING-BABY
  WEDDINGS
```

6.4.1 The Power of the "Event" Abstraction

The Electronic Scrapbook is primarily event driven. Useful information about home video is structured around events. For example, a birthday party is likely to have an "opening-presents" scene. There are probably going to be too many shots in the opening-presents scene, so the system includes a heuristic for trimming them down to a reasonable number. In contrast, the "blowing-out-the-candles" scene is important and is unlikely to need to be shortened; in a birthday party story, the system simply includes all video of blowing-out-the-candles. Story models are discussed further in chapter 8.

In practical use of the system, I have found that events are often the most accurate way to search for a remembered piece of video. Searching for "picnic" is easier than trying to remember the date and place and more accurate than searching for "food" and "outdoors."

An event is information about the function of a particular chunk of video in the lives of the participants. Knowledge about typical home video is structured around event types. This information can be used to help a home-video artist by anticipating his/her needs. The system has a vocabulary of likely events and a library of templates for stories about those events.

6.5 Actions

Actions are subtly different from events. A shot of the event "learning-to-crawl" is a specific event; the fact that the child is crawling is the subject of the video. The child may, however, perform the action "crawling" in many shots.

Actions currently used by the system are:

```
  BATHING
  CLAPPING
  COOKING
  CRAWLING
  CRYING
  DANCING
  EATING
  FIGHTING
  FROWNING
  KISSING
```

```
LAUGHING
LIFTING-CHILD
PLAYING
POSING
POUTING
READING
RUNNING
SHOPPING
SINGING
SITTING
SLEEPING
SMILING
STANDING-UP
SWIMMING
WALKING
WORKING
YELLING
```

6.5.1 Emotional Actions

To date, actions have been most useful for emotional actions such as "smiling," "laughing," "crying," and the like. It is easier for the user to make the objective judgement that subject is smiling than the subjective judgement that the shot is "happy." Emotional actions are useful for many of the functions I originally anticipated would be performed by the subjective-properties category.[6]

6.5.2 Complex Information

The representation of actions is currently simplistic. A shot merely contains an action or it does not. The subject, direct object, and indirect object of the action are not represented. This part of the representation has not been more fully developed because to date it has been surprisingly unimportant. Actions are similar to events, but events are more useful for structuring home video stories.

What are the potential benefits of a more robust representation of actions? Certainly marking the subject of actions is important and will be implemented in the future. It is useful to know that Danielle is the one who is performing the action of running. Is it useful to know that Danielle is running from the north side of the pumpkin patch to the south side? Is it useful to know that she is running away from the camera and to the right within the frame? Is it useful to know that she is running towards the parking lot and away from the other family members? How will all of this information be used? How will it be entered by the editor? How long will it take to enter? Are the benefits worth the time invested? These are unanswered questions.

[6]See section 6.9 for further discussion of subjective properties.

6.6 Significant Objects

The current significant-objects hierarchy is:

```
ANIMALS[7]
  WILD-ANIMALS
  ZOO-ANIMALS
CLOTHING
  HAT
  JEWELRY
FOOD
  BABY-FOOD
  CAKE
  JUNK-FOOD
  MESSY-FOOD
  MILK
  SPECIAL-FOOD
HOLIDAY-OBJECTS
  CHRISTMAS-TREE
  EASTER-EGG
  GIFTS
  MENORAH
  PARTY-HATS
  PUMPKIN
HOUSEHOLD-OBJECTS
  BROOM
  FORK
  KNIFE
  SINK
  SPOON
  TOILETRIES
    HAIRBRUSH
    SOAP
    TOOTHBRUSH
PLANTS
  FLOWERS
  TREES
TOYS
  BALL
  BLOCKS
  BOARD-GAME
  BUBBLE-STUFF
  DOLL
  FAVORITE-BLANKET
  FAVORITE-TOY
```

[7]The unit "animals" is actually called "animalia" and has a print name "animals." There was already a unit "animals" in the Arlotje base ontology. That representation of animals is structured differently than the representation used by the Electronic Scrapbook. The Electronic Scrapbook unit was renamed "animalia" to avoid the conflict.

The purpose of having a base ontology in a knowledge representation system is to save the user work by providing a set of basic objects. In practice, however, someone else's representation is as often an obstacle as a benefit. This is evidence for the fact that it is difficult to develop a general-purpose knowledge representation.

```
    FRISBEE
    RINGS
    SPORTS-EQUIPMENT
      BASEBALL-GLOVE
    STUFFED-ANIMAL
  VEHICLE
    AIRPLANE
    AUTOMOBILE
    BICYCLE
    BOAT
    BUS
    MOTORCYCLE
    TRICYCLE
  WATER
```

6.6.1 How Much Representation is Necessary?

As was discussed in the introduction, it would be difficult to represent every object in the frame, and a comprehensive list is unnecessary, unless you are the continuity supervisor for a feature film [Lasky 90]. In the Electronic Scrapbook, the objects category is called "Significant Objects" to attempt to convey to the user that a subjective judgement of what is important should be made.

The question then remains, should the system be able to make inferences about what objects *might* be in a scene? It could, for example, assume that there is probably water in a shot taken at the beach or in the bathroom, that there is probably a sink in the kitchen, and that there might be a turkey in a shot taken in the dining room on Thanksgiving. The ability to make these kind of inferences is something that a semantic network is useful for.

The unanswered question is in what circumstances this information is desirable. Would one ever want to do a search for "water," "sink," or "turkey"? What is the trade off between the benefits of these inferences and the cost of errors? There is not, after all, always a turkey on Thanksgiving. And while there will almost always be a sink in a kitchen, it may not be visible in the frame.

6.7 Segment Date

6.7.1 Qualitative Reasoning

Chronology is a useful organizing principle for home video subjects. Once chronology is violated, then one needs a tremendous amount of data to maintain continuity. In a fictional interactive disc made at the Media Lab in 1987 which tells the story of two people eating a spaghetti dinner, the system constantly tracks how full each person's wine glass is [Schroeder 87]. A segment of the glass being refilled is used as a transition if one wants suddenly to have more wine in the glass. With documentary subject matter, one can usually ignore these sorts of issues if natural chronology is never violated.[8]

The segment date is recorded in the form month/day/year. All date-related functions tolerate ambiguity. It is possible to omit any part of the date; a segment could be marked as "June 6th, 1987," "June 1987," "1987," or simply "June," if that is all that is known about it.

The possibility of supporting qualitative reasoning about dates was considered during the design of the system. For example, it would be convenient to give segments seasonal dates such as "Spring 1987," or relative dates like "before Christmas 1990," or "before [event x]."

However, these features become unnecessary, because in the future, it will be possible for much of the work of logging video to be done automatically. Date and time are the easiest feature to log automatically. Many currently available camcorders have clocks and can superimpose the date and time on the video. Instead of displaying it on the screen, this information could be recorded in an unused portion of the video tape such as the vertical blanking interval. There is no technological limitation to implementing this feature; it will become available when manufacturers perceive a demand for it [DSP 91].

6.8 Camera Properties

The current camera-properties hierarchy is:

```
CAMERA-SPECIAL-EFFECTS
  CAMERA-GENERATED-TITLES
  FADE
    FADE-IN
    FADE-OUT
CAMERA-MOVEMENT
  PAN
    CAMERA-FOLLOWS-OBJECT
    PAN-LEFT
    PAN-RIGHT
  TILT
    TILT-DOWN
    TILT-UP
  WALKING-WITH-THE-CAMERA
FOCAL-LENGTH
  CLOSE-UP
  EXTREME-CLOSE-UP
  MACRO-SHOT
  MEDIUM-SHOT
```

[8]Parallel editing is an interesting case in which natural chronology is not used. See section 8.5 for further discussion of parallel editing.

```
    WIDE-SHOT
    ZOOM
      ZOOM-IN
      ZOOM-OUT
  OBJECT-MOVEMENT
    OBJECT-AWAY-FROM-CAMERA
    OBJECT-TO-CAMERA
  PROBLEM-WITH-AUDIO
    AUDIO-VERY-LOUD
    AUDIO-VERY-SOFT
    BACKGROUND-NOISE
    GARBLED-AUDIO
    SENTENCE-CUT-OFF
    WORD-CUT-OFF
  PROBLEM-WITH-VIDEO
    BACK-LIT
    JIGGLE
    MAJOR-PROBLEM-WITH-VIDEO
    MINOR-PROBLEM-WITH-VIDEO
    OUT-OF-FOCUS
    TOO-BRIGHT
    TOO-DARK
  SEQUENCE-OF-SHOTS
  TRIPOD-SHOT
```

6.8.1 Video Quality

In experience with the system to date, camera properties have been most often used for indicating shot quality. If one is trying to tell a story about a child at a certain age, then there is usually too much material and one can eliminate all shots with any “problem-with-video” or “problem-with-audio.” However, if one is looking for the shot of a child’s first steps, it doesn’t really matter if the video is too dark or out of focus; the moment is important, and the quality constraints should be relaxed or eliminated.

6.8.2 Stylistic Constraints

In the original system design, I anticipated that camera information could be used to fulfill some common stylistic constraints such as avoiding jarring jump cuts and violations of the 180° degree rule.[9] However, since there is

[9]The book *The Filmmaker’s Handbook* [PA 84] provides good descriptions of these conventions: “A disconcerting mismatch between shots is called a *jump cut.* To cut from a shot of a person sitting to a shot of the same person standing in the same spot creates a disconcerting jump in time. A slight change in image size from one shot to the next (for example, two medium shots of the same person cut next to eachother) makes the cut appear to jump and feels unmotivated. A larger change in image size or angle could make the same cut in action seem natural.” (p. 280)

For the 180° rule it states that “If a subject is moving or looking right in one shot, it is best not to let screen direction change when cutting to the next shot. For example, when watching football on television, the blue team is seen moving from screen left to screen right. If the camera were now to shift to the opposite side of the field, the blue team would appear to be moving in the opposite direction (that is, their screen direction has changed from right to left.)” (p. 158) usually only one camera and there are not multiple takes of the action, in practice these stylistic problems do not often occur.

In my family's home videos, jump cuts are a serious problem in only one instance: my father interviewed my grandparents about their lives before they came to America. These interviews consist of long, static, tripod shots. Editing them creates jump cuts. However, even if the system could detect these jump cuts, there is nothing that can be done to eliminate them. The standard way to prevent a jump cut is to put something else in between -- a "cutaway" shot[10] [PA 84, p. 159]. However, there were no cutaways taken; my father's interest was in documenting his parents lives, and he did not anticipate editing the interviews. As a typical novice videographer, he was not aware of the need for cutaway shots.

In the current implementation, the system does not try to make sequences in the traditional sense. The computer may select a shot of a child falling, but it can not put together a wide shot of the child walking, followed by a close up of a tree root and the child's foot getting caught, followed by a medium shot of the child crying on the ground. The raw material to make such a sequence is rarely if ever available in typical home video. However, if the system were ever to attempt to construct a sequence depicting an event, then precise information about camera properties would be needed.

6.9 Subjective Properties

The subjective properties hierarchy is:

```
AESTHETICS
  BEAUTIFUL-SHOT
  PRETTY-SHOT
  UGLY-SHOT
ANGRY
CUTE
  VERY-CUTE
FUNNY
  VERY-FUNNY
HAPPY
INTERESTING
  VERY-INTERESTING
SAD
SENSITIVE-SUBJECT
```

[10]Some filmmakers support the use of jump cuts in this situation. Special effects such as dissolves can also be used to create transitions.

6.9.1 Lack of Perspective on the Subject Matter

By the time I logged its subjective properties, I had seen my video source material too many times to be able to decide what was cute and what was funny. Every segment already had to be either cute, funny, or somehow interesting to be included on the half-hour laser disc. It might have been easier to judge subjective properties had I marked them the first time I viewed the video.

Lack of perspective on the subject matter will probably be a common problem for home video artists. My sisters are always cute and everything my father says is funny. A normally sane colleague of mine believes that every moment of the four hours of his wedding videos is interesting. How then can one encourage users to throw things out?

When the Electronic Scrapbook is moved to a digital-video platform, users will have a limited amount of disk space to store video. Furthermore, digitizing each clip will take time. In the near future, limitations on time and storage space will force people to make choices and pare the total material down to a reasonable volume. In fact, the digitizing process will serve the same function as the need to press a laser disc served in this version.[11]

6.9.2 Sensitive Subject Matter

One subjective property which proved to be unexpectedly useful is "sensitive-subject-matter." For example, the laser disc includes video of my half sisters being born and one shot of my stepmother not fully clothed nursing the newborn Alicia. By marking these as sensitive-subject-matter, the user can decide when to exclude them from video shown.

[11]Digitizing speeds and compression ratios are rising, and the price of storage is falling. If these trends continue, in the future the user may be able to store a very large quantity of video. A key problem then arises: how can one encourage a user to throw out bad video? Experienced documentary filmmakers learn the skill of knowing when to turn the camera off; having too much raw material is a burden [Leacock 88].

7. Guessing

Logging video takes time. How much time depends on the nature of the footage and the amount of detail desired. The guess feature of The Electronic Scrapbook saves the user time by implementing a form of inheritance.

As was discussed in section 4.2, a simple hand-written log of two-and-a-half hours of home video took me seven hours to complete. This log included only basic information: people, people in audio only, location, significant objects, a rough description of camera properties, and a brief text description. It did not include a description of every object in the frame, what people are wearing, object movement, noises heard, a transcription of what was said, or precise camera information such as focal lengths, speed of pans, and the like. Furthermore, the pace of that particular video is fairly slow; shots are minutes long. In music-video style footage where shots are often only seconds long, it could take an order of magnitude more time to log an hour of video.

7.1. Level of Detail

There is a simple cost/benefit relationship between the amount of time it takes to log video versus the usefulness of the information obtained. It is important to find an appropriate level of detail. Do you need a description of every object in the frame? If you are a continuity supervisor for feature film production, you do [Lasky 90]. However, if you are a home video artist, then you probably do not.

The amount of description needed depends on the task at hand. Unfortunately, the task may be undefined or may change. Images can be reused for purposes that the logger could not anticipate.

Selecting an appropriate level of detail is not merely an issue of saving logging time. Large quantities of data can be unwieldy. The more data is present, the more difficult it often is to find the important elements. Political consultants know this problem best; the goal in preparing a memo for a politician is to summarize all the salient points in one page. Computer systems can help to solve this excess-data problem by organizing information such that the key features are highlighted, but more information is available on request.

7.2 Inheritance

7.2.1 Event-based Inheritance

My family took a trip to visit a pumpkin patch in October of 1990. All of the video from that family outing has certain properties in common: it is all from the same date, location, and event. The people present are also generally the same. Individuals appear in some shots, but not others; however, on the whole, Mari, Alicia, and Danielle are in the video and Robert is narrating.

One efficient way to log this footage is to create a long video segment for the entire event with this basic description. Shorter segments describing parts of the video more precisely can then inherit the outer description. Therefore, the user doesn't need to reenter the basic properties for each sub-segment.

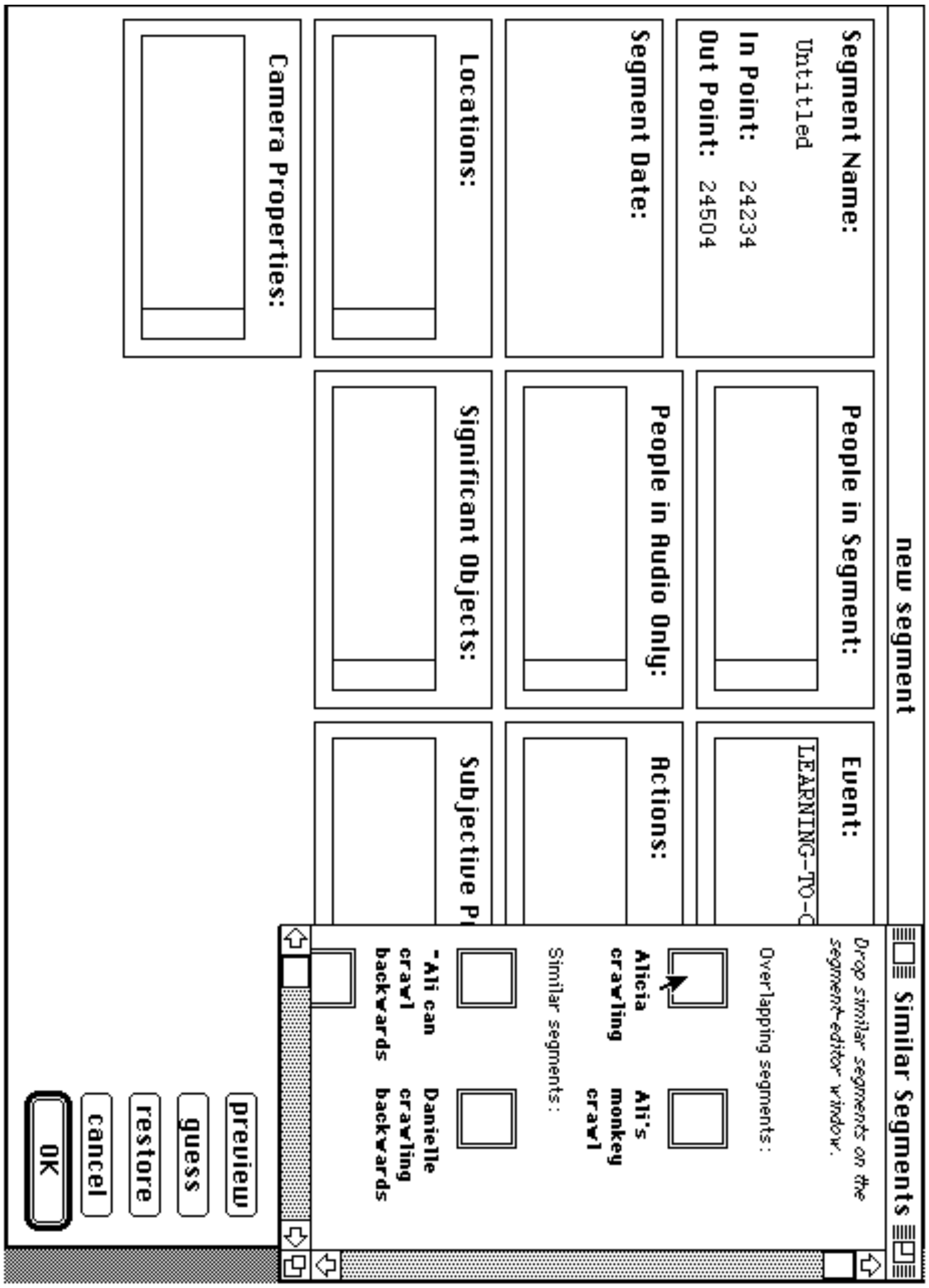
The default description can be edited for individual segments. One danger is that a user may forget to remove elements which are not present. For example, if one included the fact that Robert is present in audio only in the pumpkin patch video, it is easy to forget to remove that property if he doesn't happen to be speaking in a particular shot.

7.3 Making Inferences Apparent

It was important to me to make what the system is doing clear to the user. Rather than having inherited descriptions automatically appear, I decided to create a "guess" button on the segment-editor window.

In the original system design, clicking on the guess button added inherited values to the segment-editor window. This was charmingly mysterious -- a largely correct description simply appears, creating an impression that the system is "smart." However, the user might wonder how the system derived a guess. This interface did not provide that information.

To make the system's actions apparent, the guess button was changed; it now creates a window full of segments which overlap with the current segment and segments which are similar to it (see Figure 7.1). The function which finds similar segments will be discussed in section 7.4. To copy a description, the user picks up a segment which seems most similar to the current segment and drops it onto the segment-editor window.


new segment

Segment Name: Untitled

In Point: 24234

Out Point: 24504

Segment Date:

Locations:

Camera Properties:

People in Segment:

People in Audio Only:

Significant Objects:

Event: LEARNING-TO-C

Actions:

Subjective P

Similar Segments

Drop similar segments on the segment-editor window.

Overlapping segments:

Alicia crawling

Ali's monkey crawl

Similar segments:

"Ali can crawl backwards

Danielle crawling backwards

Figure 7.1: A segment-editor window and its corresponding "similar segments" window.

All of the values of the dropped segment are copied except for the segment name and in and out points. The date is also handled in a more complex way. A date logged can only be made more specific. A few examples of this are listed in Table 7A.

Table 7A: Copying Dates

Pre-existing Date of Target Segment	Date of Segment Being Copied	Resulting Date of Target Segment
none	12/21/65	12/21/65
1965	12/21/65	12/21/65
12/65	12/21/65	12/21/65
5/66	12/21/65	5/66
12/21	12/21/65	12/21/65

7.4 Similar Segments

If the user gives a partial description of a segment, it is possible to find segments which match that partial description. Similarity is determined by a scoring polynomial.

7.4.1 The Score Function

How does one compare apples and oranges? How is it possible computationally to say that "a shorter video segment is better, but don't make it too short, and it would be nice if took place outdoors." One approach is to use a scoring polynomial. In the scoring function used by the Electronic Scrapbook, the preceding statement would look like:

```
   (score-segments list-of-segments-to-search
      '((shortest-weighted 100 4)
        (longer-than 30 500)
        (locations-of-segment outdoors 4)))
```

The last number in each criterion is a numerical weight indicating its relative importance. If a segment is outdoors, then it will get 4 points. If it longer than 30 frames, it will get 500 points. Most segments will satisfy that criterion. If this function is used to choose among segments, this means that all segments that are shorter than 30 frames will probably be eliminated.

Shortest-weighted returns the target segment length (the second argument) divided by the actual segment length, times the weight (the third argument). (See the code below). In the example (shortest-weighted 100 4), a segment which is 100 frames long will get 4 points. One which is 200 frames long will get 2 points. One which is 50 frames long will get 8 points. One which is 1 frame long will get 400 points. The high longer-than weight was selected to offset the high scores given to short segments, since fragments are not desirable. The two criteria work together to say “make it short, but not too short.”

Any slot-value pair may be used as a scoring criterion. A list of slot-value pairs may also be given a single weight. The weight is awarded once if any one of the criteria is satisfied. Other criteria include shorter-than and longest-weighted.

The score-segments function calls the `score` function on each segment. The code for `score` is listed below:

```
(defun score (the-segment list-of-criteria)
  (let ((result 0)
        (unknown (list )))
    (dolist (criterion list-of-criteria)
      (let ((criterion-type (first criterion))
            (arg (second criterion))
            (weight (if (= (length criterion) 3)
                        (third criterion)
                        1)))
        (cond
         ((equal criterion-type 'shorter-than)
          (if (< (segment-length the-segment) arg)
            (setf result (+ result weight))))
         ((equal criterion-type 'longer-than)
          (if (> (segment-length the-segment) arg)
            (setf result (+ result weight))))
         ((equal criterion-type 'shortest-weighted)
          (setf result
                (+ result
                  (* (/ arg
                        (segment-length the-segment))
                      weight))))
         ((equal criterion-type 'longest-weighted)
          (setf result
                (+ result
                  (* (/ (segment-length the-segment) arg)
                     weight))))
         ((slotp criterion-type)
          (if
            (segment-satisfies? the-segment criterion-type arg)
            (setf result (+ result weight))))
         ((and
           (listp criterion-type)
           (slotp (first criterion-type)))
          (let  ((temp-result nil))
            (block check-slots
              (dolist (a-slot criterion-type)
                (if (segment-satisfies? the-segment a-slot arg)
                  (progn
                    (setf temp-result T)
                    (return-from check-slots)))))
            (if temp-result (setf result (+ result weight)))))
         (T
          (pushnew criterion-type unknown)))))
    (if (> (length unknown) 0)
      (dolist (c unknown)
        (format t "~&Unknown criterion: ~s" c)))
    result))
```

7.4.2 Selecting Similar Segments

To select similar segments, the system constructs a set of criteria to pass to the score function based on the existing description of the segment. Each criteria has its own default weight. These defaults are listed in Table 7B.

Table 7B: Default Weights for the Similar-Segments Function

Criterion	Default Weight
events	20
actions	10
people	4
month	3
year	3
locations	2
significant objects	2
day	1
people-in-audio-only	1
camera-properties	0
subjective properties	0

These weights were tuned to achieve useful matches. Events are particularly useful in matching segments. For example, two shots of the event `learning-to-swim` are more likely to be similar than two shots containing the object `water`.

The weights are keyword arguments to the function similar-segments. Ideally, these weights should be changeable by the user. An interface to allow this will be implemented in the future.

8. Story Models

This chapter shows how a simple "story model" can be made from combinations of functions, and how something approaching a narrative can be made with combinations of these simple story models. The results of sample story models are presented and evaluated.

8.1 Database Searches

Once a user has a database of annotated video footage, what can it be used for? The most obvious application is to perform simple database searches. For example, one might want to search for all of the shots containing a particular person, object, location, and the like. The function `segments-with-property` takes any slot-value pair. For example, one could search for:

```
(segments-with-property 'significant-objects 'cake)
```

The optional keyword argument `:segments-to-search` defaults to all known segments of the current session. This can be used to efficiently perform a logical "and." For example, the following statement finds all segments of cake at a picnic:

```
(segments-with-property 'events-of-segment 'picnic
   :segments-to-search
     (segments-with-property 'significant-objects 'cake))
```

Other simple search functions include `segments-without-property` and `segments-with-property-or`. The following statement finds all segments either at the beach or containing water which do not have a problem with the video quality:

```
(segments-with-property-or
   '((significant-objects 'water)
     (locations-of-segment 'beach))
   :segments-to-search
     (segments-without-property
         'camera-properties 'problem-with-video))
```

8.1.1 Checking the Knowledge Hierarchy

Note that searching for `problem-with-video` will also return subsets of `problem-with-video` such as `too-dark`, `out-of-focus`, and the like. Similarly, searching for `toys` will return `dolls`, `bubble-stuff`, and `frisbees`. The system checks for subsets unless this option is disabled by setting the keyword argument `:check-subsets` to `nil`. This implements a useful kind of inheritance.

8.1.2 Searching for People

In videos of children, it is often desirable to look for all of the segments of a child at a particular age. It is also convenient for the system to convert between a person's name ("Alicia") and the unit name (`humans.1`). The function segments-with-person-at-age takes the arguments months, days, years, to-months, to-days, and to-years. If the end date is not specified, then it is assumed to be one plus the most-specific start date specified. For example, the following statement finds all shots of Alicia at age 0 to 1:

```
(segments-with-person-at-age "alicia" :years 0)
```

Whereas this statement finds all shots of Alicia at age 18 to 19 months:

```
(segments-with-person-at-age "alicia" :years 1 :months 6)
```

Ages are determined by examining segment dates and the individual's birth date. Variable month lengths and leap years are properly accounted for.

Writing lisp code to perform simple searches and combinations of searches is easy; however, designing and writing interface code to make this functionality available to the user is a challenge. This task is left for future work.

8.2 Simple Story Models

In the current data set, a search for all segments with Alicia at age 0 to 1 years returns 36 video segments. This list is useful as raw source material for making a more polished presentation; however, it not pleasant to watch from start to finish. The list of shots is in random order, contains segments that have technical problems (such as dark video or garbled audio), contains sensitive subject matter (such as a child being born), and contains overlapping segments.

In the Electronic Scrapbook, a simple `story-model` consists of a database search function and a set of criteria for how to process the raw list. The goal is to transform a long, unorganized list into something that is enjoyable to watch in sequence.

8.2.1 The Basic Search

A `story-model` is an Arlotje object with slots that hold functions. The function `run-story-model` executes those functions and returns a list of segments:

```
(defun run-story-model (my-story-model &rest args)
  (if (compound-story-model-p my-story-model)
    (apply 'run-compound-story-model my-story-model args)
    (progn
      (if (not *running-compound-model*)
        (setf *discards* nil))
      (let* ((initial-list
              (apply
               (get-value my-story-model 'story-model-function)
               args))
             (result-list  nil)
             (sort-function
              (get-value my-story-model 'sort-function))
             (after-sort-function
              (get-value my-story-model 'after-sort-function)))
        (setf result-list
              (remove-segments
               initial-list
               (get-value my-story-model 'remove-criteria)
               (get-value my-story-model 'remove-threshold)))
        (setf result-list
              (remove-overlaps
               result-list
               (get-value my-story-model
                          'remove-overlaps-criteria)))
        (if (and sort-function
                 (not (failurep sort-function)))
          (setf result-list (apply sort-function result-list args)))
        (if (and after-sort-function
                 (not (failurep after-sort-function)))
          (setf result-list
                (apply after-sort-function result-list args)))
        result-list))))
```

The slot `story-model-function` holds a database search algorithm used to generate the initial list of segments to be manipulated.

8.2.2 Discards

Any segments later removed from this initial list are added to the list `*discards*`, which is reset at the start of a new `story-model`. Discarded segments can be examined by the user by clicking on the "discards" button (see Figure 8.1). The user can recover segments from the discards window simply by dragging them back into the scrapbook window. Segments can also be rearranged or discarded by being dragged to the trash window. All decisions made by the system can be changed; creative control remains in the hands of the user.

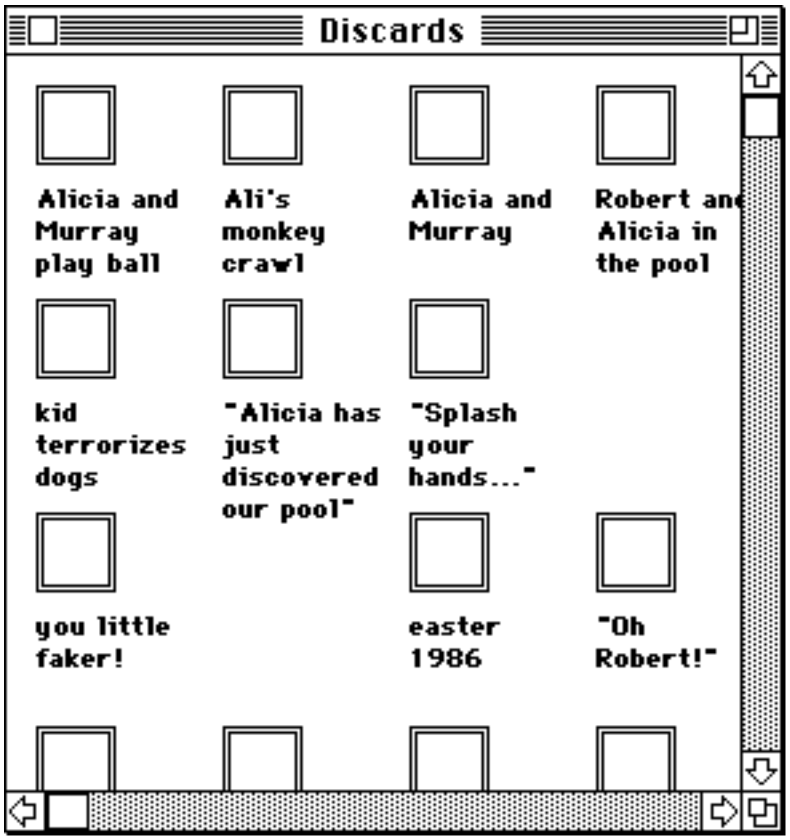


Figure 8.1: The discards window.

8.2.3 Filtering

Segments with undesirable properties can be removed by the function `remove-segments`. Each `story-model` has a slot `remove-criteria`. These criteria are passed on to the function `score`.[1] All segments with a score higher than the `remove-threshold` are discarded. The definition of the slots `remove-criteria` and `remove-threshold` and their default values are:

```
(define-unit remove-criteria
  (works-like 'prototypical-slot)
  (makes-sense-for 'story-model-objects)
  (value-defaults-to
     '((camera-properties major-problem-with-video 101)
       (subjective-properties-of-segment
          sensitive-subject 101)
       (camera-properties minor-problem-with-video -35)
       (camera-properties problem-with-video 51)
       (longer-than 3000 51)
       (longer-than 4000 51))))
```

[1]See section 7.4.1 for more discussion of the score function.

```
(define-unit remove-threshold
  (works-like 'prototypical-slot)
  (makes-sense-for 'story-model-objects)
  (value-defaults-to '100))
```

These defaults can be paraphrased as "discard anything which is sensitive subject matter. A long segment is bad and a very long segment is very bad. Discard anything with a major problem with the video; ignore minor problems with video." Since these criteria combine, a segment which is long but not very long (3000 to 3999 frames) will be kept unless it also has a problem with the video.

These criteria can be changed for individual `story-models`. For example, in the `important-firsts story-model`, the video quality criteria are eliminated -- if one is looking for shots of a child's first steps, it is less important if the video is dark or out of focus.[2] Similarly, if one wants to watch a child being born, then sensitive subject matter should not be filtered out. The stylistic criteria are content-dependent.

8.2.4 Removing Overlapping Segments

It is undesirable to show the same video multiple times in the same story, unless one has deliberately chosen to do so. The system can help the user by detecting overlapping segments and using the scoring function to choose between them. The default is to choose the shortest segment which is not too short. The scoring criteria are kept on the `remove-overlaps-criteria` slot of the `story-model`:

```
(define-unit remove-overlaps-criteria
  (works-like 'prototypical-slot)
  (makes-sense-for 'story-model-objects)
  (value-defaults-to '((shortest-weighted 30)
                       (longer-than 30 31))))
```

This feature is quite useful, but the length-based selection criteria are unsatisfying. Consider, for example, this sequence:

(In-point	*one.)*
Mari:	Let's stand up. [Help's Alicia to stand] Aaaw! We're so strong. We're so big and cute!
Rose:	Is that a pinafore or part of the dress?
Mari:	It's a pinafore. It's part of the dress. Give Mommy your hand.
(In-point	*two.)*
Robert:	Don't help her stand. Put her down.
Mari:	[Holds Alicia by one hand.] She's in third position. [Everyone laughs.] Look at her feet!

[2]See section 8.5 for more discussion of the important-firsts story model.

Robert: That's a riot.
[Alicia falls.]
Mari: Oh, look at that face.
[Alicia begins to cry.]
(Out-point two.)
[Mari picks up Alicia. Camera zooms in close.]
[Alicia stops crying as she is picked up.]
Mari: Smile for Daddy! [Alicia smiles.] That is a big smile. [Mari tickles Alicia, lifting her up and down. Alicia giggles.] You little faker! You little faker!
(In-point three.)
[Mari continues to tickle Alicia. Both smile and laugh.]
(Out-points one and three.)

The long version of this segment tells a story. The impact of the story is different if one begins with Robert saying "don't help her stand" and ends with Alicia crying. On the other hand, if one takes merely the shot of Alicia and Mari laughing and giggling, then the meaning is entirely different -- the story of Alicia trying to stand is eliminated.

The close up of Mari tickling Alicia is a beautiful shot. If one used subjective judgements of quality to select video, then that last segment would be chosen over the others. However, that criterion is as unsatisfying as segment length.

The complete segment is 55 seconds long. Substantially different meanings can be made from this video, depending on exactly which chunk of it is used. How is a computer to choose between them? One might first ask, how does a person choose between them?

8.2.5 Sorting

After overlapping segments are removed, the system sorts the remaining list. The default sort function is chronological:

```
(define-unit sort-function
  (works-like 'prototypical-slot)
  (makes-sense-for 'story-model-objects)
  (value-defaults-to
   #'(lambda (list-of-segments &rest args)
       (declare (ignore args))
       (sort (copy-list list-of-segments) #'in-point-less-than-p))))
```

In the `kids-at-same-age` `story-model`, the video is sorted by the age of the children. Instead of showing Alicia and Danielle in 1986, it shows them each at the same ages.[3]

[3]This story model and the important-firsts story model (see the next section) were designed with positive goals in mind: "wouldn't it be nice to see both kids just learning to crawl?" These

```
(define-unit kids-at-same-age
  (member-of 'story-model-objects)
  (story-model-function
   #'(lambda (&key
              (list-of-kids nil)
              &allow-other-keys)
       (let ((list-of-criteria (list )))
         (dolist (child list-of-kids)
           (pushnew (list 'people-in-segment
                          (human-name-to-unit child))
                    list-of-criteria))
         (segments-with-property-or list-of-criteria))))
  (sort-function
   #'(lambda (list-of-segments &key
                               (list-of-kids nil)
                               &allow-other-keys)
       (age-sort list-of-segments list-of-kids
                 :up-to-second-oldest T)))
  (after-sort-function
   #'(lambda (list-of-segments &key
                               (list-of-kids nil)
                               &allow-other-keys)
       (person-pattern
        list-of-segments list-of-kids
        '((camera-properties major-problem-with-video -101)
          (subjective-properties-of-segment sensitive-subject -101)
          (camera-properties minor-problem-with-video 35)
          (camera-properties problem-with-video -51)
          (longer-than 3000 -51)
          (longer-than 4000 -51)
          (shortest-weighted 30)
          (shorter-than 30 -51))))))
```

The results of this `story-model` will be discussed in section 8.5.

8.2.6 One Last Pass: the After-sort Function

It is sometimes desirable to process a list further after it has been sorted. This is performed by the `after-sort-function`. The default is for no `after-sort-function` to be used; however, the code in the previous section provides an example of how it can be used. After the video is sorted by the age of the children, it is desirable to assure that the list alternates between children. This is accomplished by the **`person-pattern`** function, which works rather like the `remove-overlaps` function. When a sequence of more than one shot of the same child is detected, a scoring polynomial is used to choose between them.

models have the unfortunate side effect, however, of encouraging the viewer to competitively compare children's accomplishments.

8.3 Compound Story Models

A `story-model` can be made up of other `story-models`. With this feature, one can begin to construct narratives. The `important-firsts story-model` is made up of key events in a child's development:

```
(define-unit important-firsts
  (member-of 'compound-story-models)
  (components '(kids-birth kids-bottle-fed kids-learning-to-sit-up
               kids-burped kids-learning-to-crawl-backwards
               kids-learning-to-crawl kids-first-solid-food
               kids-learning-to-stand-up kids-learning-to-walk)))
```

Each of the component `story-models` can themselves be made up of other `story-models`; any level of depth is possible.[4]

8.4 Results of a simple story-model: `child-at-age`

In the current data set, a simple database search for all shots of Alicia at age 0 to 1 years returns 36 segments:[5]

```
Alicia Being Born (long)
"Ali, this is your Dad"
Alicia just born
close up of Alicia as a newborn
Mari nursing Alicia
hiccups
Alicia and the rings (long)
rings in the mouth
"Oh Robert!"
Enid feeding Alicia
Christmas 1985
returning from a hike
Alicia eats solid food
Easter 1986
you little faker!
"Smile for Daddy!"
"She's in third position"
Mari tickling Alicia
Alicia's dexterity
Alicia discovers the pool (long)
"Splash your hands..."
"Alicia has just discovered our pool"
"Swim to Daddy!"
"Ali can crawl backwards"
```

[4]The `run-story-model` function is recursive. If a `story-model` is compound, then `run-story-model` calls itself on each of the parts.
[5]The titles of the segments were subjectively, rather than systematically selected -- I picked something that would remind me of the segment. Consequently, these titles may not be meaningful to the reader. There is not enough space here for complete segment descriptions; the titles should convey a general sense of the results.

```
kid terrorizes dogs
Where are the puppies?
"Two trusty dogs."
Robert and Alicia in the pool
"Swim swim swim!"
Alicia and Murray
"Wanna go for a ride?"
Ali's monkey crawl
Alicia crawling
Alicia and Murray play ball
"This game is getting tiring."
Alicia squeezes cake
```

The total running time of the list is 18.6 minutes.

The `story-model` "child-at-age" applies the default `story-model` filtering and sorting functions to the same base search function:

```
(define-unit child-at-age
  (member-of 'story-model-objects)
  (story-model-function #'(lambda (&key
                                   (kid nil)
                                   (years nil)
                                   (months nil)
                                   (days nil)
                                   (to-years nil)
                                   (to-months nil)
                                   (to-days nil)
                                   &allow-other-keys)
                            (funcall 'segments-with-person-at-age
                                    kid
                                     :months months
                                     :years years
                                     :days days
                                     :to-months to-months
                                     :to-years to-years
                                     :to-days to-days))))
```

Running this `story-model` with the same arguments, Alicia at age 0 to 1 years, returns a list of 20 segments:

```
"Ali, this is your Dad"
close up of Alicia as a newborn
hiccups
rings in the mouth
Enid feeding Alicia
returning from a hike
Alicia eats solid food
"She's in third position"
"Smile for Daddy!"
Mari tickling Alicia
Alicia's dexterity
"Swim to Daddy!"
"Ali can crawl backwards"
Where are the puppies?
"Two trusty dogs."
```

```
"Swim swim swim!"
"Wanna go for a ride?"
Alicia crawling
"This game is getting tiring."
Alicia squeezes cake
```

Four segments are eliminated by the remove-segments function and twelve are eliminated by remove-overlaps. The total running time of the resulting list is 5.2 minutes, 70% shorter than the original.

The result is surprisingly coherent -- it begins with Alicia being born and progresses to her first birthday party. The shots selected are reasonably representative.

There is, however, one jump cut. The shots "Smile for Daddy!" and "Mari tickling Alicia" are both close ups of Mari holding Alicia, and the transition is awkward. To prevent this, the system could be made to check for two shots together that have the same focal length, location, and people present. However, without precise camera angle information, this might eliminate acceptable transitions.

8.5 Comparison of two approaches: `kids-at-same-age` and `important-firsts`

Video does not always need to be presented chronologically. For example, it is interesting to watch a group of siblings at the same ages rather than the same years -- instead of showing Alicia and Danielle in 1990, one can show Alicia and Danielle at 18 months old. As discussed in section 8.2.6, the `kids-at-same-age` model sorts video by the age of the kids. In cinematic terms, this uses the *parallel action* model of editing.[6] The results of the `kids-at-same-age story-model` for Alicia and Danielle are:

```
"Daad!  Enough!"
close up of Alicia as a newborn
Danielle and Baby Mickey
returning from a hike
Danielle crawling backwards
"Smile for Daddy!"
"What's so funny?"
"Swim to Daddy!"
"Danielle, come feed the goats."
```

[6]Parallel editing is a fundamental cinematic technique. The first known example is a 1903 black and white, silent, short by Edwin S. Porter called *The Life of an American Fireman*. See [Cook 81].

An alternative, more knowledge-based approach is to show the children at key stages in their development such as learning to crawl or eating their first solid food. This is done by the `important-firsts story-model`:

```
close up of Alicia as a newborn
Danielle as a newborn
"Ali can crawl backwards"
Danielle crawling backwards
Alicia crawling
Danielle crawling
Alicia eats solid food
"She acts like she eats it all the time!"
```

This presentation has more structure and comes closer to being a “narrative.” The difference highlights the benefits of a knowledge-based approach. Selection of segments in this model is made more often based on content than stylistic criteria, and the result has greater coherence.

The `kids-at-same-age` model for this data has nine segments and is 2.1 minutes long. It is interesting to note that the `important-firsts` model has fewer segments but is longer: it has eight segments and is 2.9 minutes long. This is indicative of the fact that the `important-firsts` model makes less use of stylistic criteria. The stylistic criteria are weighted to give preference to shorter segments.

The `important-firsts` model does, however, contain a few dark shots. The video quality criteria were removed from this `story-model` (and from its component `story-models`.)[7] If one is looking for a shot of Junior’s first steps, it is less important that the shot might have technical problems -- it is an important moment. On the other hand, if one is merely looking for any shot of junior at age 3 months, then video and audio quality are useful selection criteria.

It could be argued that the greater randomness of the `kids-at-same-age` model is more interesting than the more formulaic `important-firsts` model. Certainly both ways of telling this story have merits. However, randomness is a poor substitute for inventive combination.

A higher goal would be for the system to make “creative” suggestions. However, the term “creativity” is as ill-defined as the term “intelligence.” Humans often define “creativity” and “intelligence” as everything computers can not do. To avoid using these terms, one might say that the system will always benefit from additions to its vocabulary of ways to combine video based on both stylistic and semantic criteria.

[7]See section 8.3 for more discussion of compound story models.

9. Related Work

This chapter traces the system's roots in computer science and artificial intelligence.

9.1 Case-Based Reasoning

The original design for The Electronic Scrapbook was inspired by research on Case-based reasoning (CBR) and case-based planning (CBP). As the design evolved, it came to be something different, which I will call "knowledge-based templates."

In the traditional definition of Case-Based Reasoning (CBR), the computer is first given a description of a starting state. It retrieves a case from memory which is similar to the problem at hand, adapts that case to the current situation, and applies it [KS 84; RS 89].

In editing film or video material, describing the starting and goal states is the core of the problem. The starting state is raw, undescribed, unedited footage (rushes). The goal is a polished, edited presentation.

The story models used in The Electronic Scrapbook are similar to cases. Each story model tells the computer how to deal with a specific situation. One case in Kristian Hammond's "Chef" case-based planning program[1] tells the system how to make beef with green beans [RS 89]; a case in The Electronic Scrapbook tells the system how to tell the story of two siblings growing up.

In traditional CBR, the system selects a case and adapts it to the situation at hand. In The Electronic Scrapbook, the user performs the actions of case selection and case adaptation. The user selects a story model and runs it to generate a rough cut. The user may then modify the rough cut directly or modify the story model used to generate the rough cut and try the model again. The process is iterative and cooperative between the system and the user.

[1]The differences between case-based reasoning and case-based planning can be subtle. Janet Kolodner's Cyrus program is an example of case-based reasoning: the system reads stories about the life of Cyrus Vance and answers questions. The Chef program is an example of case-based planning: the system plans how to get from a starting state (raw ingredients) to a goal state (a finished dish). In CBR, the goal of the system is the result (answers to questions); in CBP, the goal is how you reach the result (the recipe).

The Electronic Scrapbook is more similar to CBR than CBP; the final video rather than the process of generating the video is the goal. However, it should be noted that enjoying the act of creating the video is a fundamental goal of the system. In that sense, the distinction between CBR and CBP begins to blur.

While the details of these processes are different, the most important similarity is that they are both knowledge-based -- a library of specific domain knowledge is used to solve the problem at hand. The most important difference is that in the Electronic Scrapbook, the user is a partner in the creative process.

9.2 Text-Based Story Generation

9.2.1 Manipulating a Limited Vocabulary

The essential problem of video editing is to select and order a subset of the limited quantity of available material. This is similar to work that has been done in text-based story generation, which begins with a limited vocabulary [Meehan 81]. One difference is that the starting video material is incomplete and imperfect -- one may not have a shot of the family piling into the car on the way to the beach, even though this is an easy way to start a story about a trip to the beach. The challenge is to make the best of the available material.

Another difference is the addition of an extra step: the need to describe the starting material. This is currently accomplished by helping the user to transcode from video to a semantic representation. In the future, more of this work will be accomplished by having the computer examine the video signal directly. Information can also be gathered at the camera and recorded in unused portions of the video tape such as the vertical-blanking interval [DSP 91].

Grammatical and rhetorical rules are somewhat but not entirely analogous to stylistic constraints of video editing.[2] Most of the conventions of video editing are stylistic guides and not firm rules. Video provides an opportunity to explore the concept of style of presentation more easily than text.

9.2.2 Goals

Many story-generation systems are goal driven. In James Meehan's Tale-Spin system, the goals of the characters drive the process of creation of a story: Wilma bird flies to the river because she is thirsty [Meehan 81]. In Natalie Dehn's AUTHOR system, the goals of a computer-simulated author shape the story: the protagonist's mother falls ill to enhance the drama of the story [Dehn 89]. In the Electronic Scrapbook, the system anticipates the goals of a typical human author: it is likely that a home-video artist would want to tell a story about a child growing up, so a story model on that topic is provided.

[2]The film theorist Christian Metz formally tackled the question "In what ways and to what extent is cinema like verbal language?" See the discussion of Metz's work in [Andrew 76].

9.3 Scripts

As was discussed in section 6.4, events in the Electronic Scrapbook are similar to Roger Schank's work on scripts [SA 77]. For example, a story-model could be created based on Schank's famous example of eating in a restaurant. Eating-in-a-restaurant is an event with component scenes ordering, eating, paying, and the like.

In Schank's work, each slot in the script is filled once. In working with video material, there may be elements duplicated and elements missing. For example, there may be thirty minutes of video showing Ken eating a fish dinner. How much of this video should be used? The Electronic Scrapbook uses stylistic constraints to choose between segments.

Eating in a restaurant is an example of a highly-structured event. One almost always orders food and pays a check (either before or after eating), and these likely scenes can be used to structure a story about this event. In this example, stylistic constraints are used to cut down on the total amount of video used.

In contrast, an event like `child-playing` is less structured. It has no likely scenes that can help structure the story. In this example, the stylistic constraints play a central role in shaping the resulting story. Events and stylistic constraints are complementary; each story model has its own balance of semantic and aesthetic concerns. Stylistic constraints extend the power of scripts.

9.4 The Cyc Project

The idea of creating a representation of everyday life was inspired in part by the Cyc project, an effort to create a large "common sense knowledge base" at the Microcomputer Consortium (MCC). Doug Lenat, leader of the project, believes that computer representations are brittle because computers lack "common sense." He defines common sense as the myriad facts about ordinary life which we usually take for granted: that dogs usually have four legs, and that breathing is a bodily function which is a "Composite Physical & Mental Event Type" [LG 90, p. 32]. A team of engineers is attempting to represent "consensus reality" as part of this ten-year project.

Will something like "common sense" emerge when Cyc has a critical mass of information? This is an open question. The ambitions of The Electronic Scrapbook are much more modest in scale. The "suburban ontology" is a special-purpose knowledge representation designed to aid in the performance of a specific task.

10. Future Work

10.1 More Data

10.1.1 Birthday Parties

To date, the system has been tested with only one data set. It would be beneficial to see how it performs on a variety of source material.

The idea of editing video of a child's birthday party was used as a thought experiment during the process of system design. It would be interesting to see how the system performs on this structured information.[1]

10.1.2 Scaling

It would also be beneficial to work with a much larger data set. Some of the power of the underlying knowledge representation is currently unnecessary, because half an hour of video is manageable without it. The larger the volume of data, the more the system's design will be tested.

10.1.3 Storage, Memory, and Speed Issues

There are technical limitations which currently prevent using a larger amount of data. Right now the system is limited to half an hour of video, the amount that will fit on one side of a laser disc. If the system is converted to digital video, then the limitation will be the amount of available hard drive storage space.

The amount of memory is also an issue. All data is currently loaded into memory. When lisp runs out of memory, it stops to garbage collect. Garbage collection takes approximately 30 seconds on a Macintosh IIx computer. With 30 minutes of video data loaded, the system currently stops to garbage collect on every significant action. For example, it garbage collects every time the user clicks on the "guess" button or runs a `story-model`. Using a faster computer with more memory can alleviate this problem. Ephemeral garbage collection,[2] which has just recently been implemented in MCL, will also ease this problem.

Significantly less memory would be required if information were stored in a relational database rather than in memory. However, searching and inferencing mechanisms access a large amount of data. Repeatedly getting

[1]See section 4.1 for an explanation of the use of the phrase "structured information."
[2]With ephemeral garbage collection, certain kinds of memory can be freed for reuse in the background without interrupting the user.

data from disk would be slow. Therefore, it is desirable to keep as much information in memory as possible.

A number of improvements can be made to the software to increase its speed. It seems likely that the system is being slowed down by constantly recreating list browsers and hierarchically sorting their contents. These could instead be recycled. The system would need a mechanism for detecting when a list needs to be updated. For example, the system currently makes a new hierarchically-sorted list of possible locations every time a segment is created. It would be preferable to use the same list over and over, and recalculate it only after a new location has been created.

However, it is only a speculation that remaking list browsers is slowing the system down significantly. The best way to improve the speed of the code is to run a speed-checking program and find out exactly which functions are using up most of the system's time.

The underlying knowledge representation language is part of the culprit. The author, Kenneth Haase, promises that his next version of Arlotje will be faster.

10.2 Digital Video

The system was designed to be used with digital video. With the release of Apple Computer's QuickTime™ digital video software, this has become more practical. Preliminary work on this conversion is under way.

10.2.1 Still Frames to Represent Segments

First and most simply, it would be desirable to have each video segment represented by a digitized still frame of video. The icon for video segments is an empty picture frame; this frame is meant to be filled (see Figure 10.1).

Figure 10.1: A video segment.

10.2.2 A Linear Representation of the Source Material

If the user is to digitize video from tape, then it will be necessary to write a driver for the tape deck and an interface for digitizing chunks of video. This interface would benefit from a graphical, linear representation of the source tape.

Users could mark in and out points of properties on this linear representation, and the properties would be inherited by all video segments in the selected range. For example, if entering the park is marked at frame 20100 and leaving the park is marked at frame 24500, then any video chunks selected from that range are assumed to be in the park.

This is analogous to the process currently used in the Electronic Scrapbook in which a user makes one segment for the entirety of our trip to the park, and all overlapping segments can inherit that description.[3] In a linear timeline representation, however, each property can be inherited separately. Furthermore, the process of inheritance can be more easily visualized by the user.

If a great deal of description is to be done on a timeline, then it will get too crowded to read. Ideally, the timeline should be expandable to multiple layers or "tracks." For example, one track could show who is present and another the actions being performed. The audio could have a separate representation from the video. Particular tracks could be combined, separated, shown, or hidden [Davis 91].

10.3 An Interface to Story-Models

There is not yet an interface to story-models. Story-models are currently executed by typing lisp code. Designing an interface to story-models is essentially an issue of visual programming [Shu 88]. Visual programming is an entire field of research. What would such an interface look like? Some preliminary ideas are presented here.

First, the user needs a list of available story-models. Names of story-models will not be sufficient to communicate what each model does. It would be useful for a text description to pop up when a model is highlighted.

A dialog box for simple story-models must show the functions used by the model. How are these to be displayed? Scoring criteria such as `'((shorter-than 300 50) (subjective-properties sensitive-subject-matter 101))` are easier to understand than lisp code, but still require significant explanation. Is

[3]See section 7.2.

there any simpler way to present this information and give the user the ability to modify it?

An interface is needed to help the user to construct the base search function. Different levels of assistance can be provided to the user. The user could be required to write lisp code. Alternatively, the user could be provided with a database query language which would then be transformed into lisp code. At an intermediate level of assistance, the user might be guided through the process of constructing a query with carefully designed dialog-boxes and menus. Ideally, however, the user should be provided with a graphical programming language in which queries can be easily constructed and understood.

The task of making an interface to story-models raises fundamental questions of human-computer interface design. A simple interface requiring significant knowledge on the part of the user will be constructed in the near future. The task of designing a visual programming language is an open research issue.

10.4 A Graphical Interface to the Underlying Knowledge Representation

The need for a graphical representation to the underlying knowledge representation was discussed in section 6.1.3. A graphical representation would enable the user to visualize the knowledge base and watch inferences being made. A graphical representation is also needed to log complex data structures such as actions. For example, the action "walking" should be given a person, a starting location, an ending location, and perhaps other modifiers. Consider representing a video clip in which Cynthia walks briskly from the beach blanket to the water. One might further need to represent the fact that the blanket is on the sand, which is part of the beach. The representation becomes more complex if one wants to indicate why she walked to the water--to rescue Amy's ball from the waves.[4] Such a representation has many levels, and is best displayed and manipulated graphically.

Three different graphical representations are envisioned. The first is a planar representation in which units are connected by arrows showing their relationship (see Figure 2.1). The second is a timeline showing how the linear source material changes over time (see section 10.2.2). A timeline could also be used to represent events in real time. Each of these schemes meets different needs. How they can be combined into a coherent whole is an open question.

[4]The issue of how much knowledge representation is necessary was discussed in sections 2.4 and 6.6.1.

Future versions of Arlotje will come with a graphical interface [Haase 91]. This will reduce the amount of interface coding necessary to construct an application with Arlotje, and it will also help programmers to understand its workings.

10.5 Making the Interface Reconfigurable

The interface and underlying representation could be made reconfigurable by the user. This would make the logging system useful for new subject domains. For example, if one were making a scrapbook about the zoo, one might want to add a pane to the segment-editor window for animals, separating them from objects or people. An interface to enable the user to make that kind of change could be made which would not require much knowledge on the part of the user. Object-oriented programming makes this possible; one can define a new instance of `list-pane`, and the system will do all the work of creating a corresponding `list-browser` and the like.

It is perfectly feasible for the user to redefine the base ontology. The user could select which ontology to load: "home video 1," "home video 2," "zoo," "vacation 1," and the like. The challenge comes in the design of story-models. Story-models currently incorporate specific domain knowledge. The model `important-firsts` looks for specific event categories like `learning-to-crawl` and `first-solid-food`. Strategies for managing specific events are built into the system. This is how the system helps to anticipate the needs of the user. While the programmer might provide a different set of story models for the "vacation" ontology, it would be difficult for the user to construct these without programming.

10.6 Annotations

10.6.1 Titles

It is feasible for story models to automatically generate titles for their scrapbook page. For example:

```
(run-story-model ‘child-at-age :kid “Danielle” :years 2)
```

The above model could generate the title "Danielle at age 2." The `important-firsts` model for Alicia and Danielle might generate the title "Important Moments for Alicia and Danielle." Since each story-model can have its own method for creating a title from its arguments, title generation poses no technical problems.

Users, of course, would need to be able also to generate and edit titles by hand. Graphically editable objects like Hypercard fields would be ideal.

10.6.2 Histories

The system also needs to provide documentation for how a particular `story-model` was generated. If users are to learn about the system and about video in general by running `story-models` multiple times, then they must be able to look up how a particular story was generated -- what `story-model` if any was run, what arguments were used, and how it was subsequently modified by the user.

10.6.3 Why

Documentation would be useful down to the level of individual segments. Why was this segment included in this list? The answer will help to explain the workings of the system to the user, and may help to detect errors in the representation. For example, suppose the user sees a segment which seems out of place and selects the "why" button. The system responds that the shot was included because it contains Alicia at age three; however, the user notes that Alicia is not actually there -- the shot was incorrectly logged.

A "why" feature would be particularly helpful if the system were given more ambitious methods for sequencing shots. Why is this close up of the sofa here? Because the system detected a jump cut and was searching for a shot of an object in this same room to act as a transition.

10.7 Scrapbook Page Types

Story-models are currently used to help find and arrange logged footage. This process can be inverted: the system's knowledge about typical story types could be used to help log footage.

One approach is to give the system a set of specialized scrapbook page types which make inferences about segments put on them. For example, creating a "child-at-age" page for Danielle at age 1 would automatically put the title "Danielle at age 1" on the page. It can then be inferred that every video segment placed on that page contains Danielle at that age. Similarly, one could create a "vacation" page for our trip to Italy in 1989. Every segment placed on that page would be assumed to be from that time and place, unless different information is given.

The key problem that again arises is how to inform the user that inferences are being made. For example, one might place the segment "getting on the plane" on the page "our trip to Italy," and while it was shot in 1989, it did not take place in Italy. How is this error to be prevented?

It would be possible for the system to create a dialog box asking the user to confirm each inference. However, the constant interruptions would make the feature more of a nuisance than a convenience.

Another approach would be to create an inferences window in which text descriptions of inferences are sent. The system might write that "I am assuming that segment 'Amy buys a leather jacket' took place in Italy in 1989." This would be less intrusive, because the user is not interrupted.

There is the danger, however, that the user might not notice these messages. If a large number of inferences are being made, the resulting stream of text is likely to be ignored. One way to reduce the number of such messages is to only notify the user of an inference when it is *used* as part of a retrieval or composition process. There is a trade-off between the convenience of inferences and the cost of errors.

10.8 Object Permanence

10.8.1 Instances

In the current implementation of the Electronic Scrapbook, when one selects the object "sofa" for a particular video segment, the unit `sofa` is placed on the segment's `significant-objects` slot. `Sofa` is the collection of all sofas. One has effectively said "there is a sofa in this shot."

It would be possible instead to create a specific instance of the collection `sofa`. The advantage is that this instance could then be annotated -- all sofas are not pink, but this specific sofa is pink. The unit would be called something like sofa.4 and its `print-name` might be "the pink sofa by the fireplace." `Sofa.4` would have a `my-color` slot holding the value `pink`. Other slots could record its typical location and its typical position in that location.[5]

When a user indicates that there is a sofa in a particular video segment, the system could try to figure out if it is a known sofa. If it is in the living room in Mill Valley, then it is likely to be "the pink sofa by the fireplace." If the total amount of data in the system is small, then the user could be asked to select from a list of all known sofas.

[5]Sofas have usual locations, most houses have fixed locations, and frisbees have no fixed location when in use, but may have a usual location when not in use (depending on how neat the owner is.) These details would need to be represented in order to track individual objects.

By tracking specific instances of objects, the system is able to accumulate knowledge. One might search for everything that is pink and this would return Alicia's favorite blanket, Danielle's tricycle, the sofa in the living room, the hand towels in the guest bathroom, and the tiles at the entrance of the Duomo in Florence. This level of detail of representation is only practical if the system tracks specific objects.[6]

Unfortunately, selecting which if any of the known sofas for the new instance of sofa is non-trivial. "The sofa in the living room at home" is different before and after July 1987, because the family moved and bought new furniture. It is possible to create to separate instances of sofa, `sofa.4` and `sofa.6`, and later obtain new information which indicates that they are really the same. The two units would need to be merged. Similarly, the system might infer that the sofa in two segments were both `sofa.4`. If it is later discovered that the sofa in one segment has small gray flowers on it, then the question arises: does the sofa in the other segment also have small gray flowers, or are these really two different sofas?

10.8.2 Temporal Extents

To say that "there is a sofa in this segment" is very general. One can get more specific by saying that "`sofa.4` is in this segment." However, even that is a generalization. Strictly speaking, there is a *temporal extent* of `sofa.4` in this segment, `sofa.4` at 11:50 am to 11:51 am on July 12th, 1991. This allows one to represent the fact that `sofa.4` in 1987 is brand new, but `sofa.4` in 1991 has a coffee stain on the arm. Similarly, one can not simply say that Alicia is 3′ 6″ tall. Alicia in August of 1991 is 3′ 6″ tall, but Alicia in August of 1986 is only 2′ tall.

[6]Whether this degree of detail is *desirable* is discussed in sections 2.4 and 6.6.1.

11. Conclusion

11.1 Interface Design

The Oxford English Dictionary traces the origin of the word “interface” to a technical text from 1882 which states that “The term interface denotes a face of separation, plane or curved, between two contiguous portions of the same substance” [OED 71]. The modern sense of the word is traced back to an article in the Washington, DC *Evening Star* in 1962 which states that “Interface... seems to mean the liaison between two different agencies that may be working on the same problem”[1] [OED 87].

It is tempting to view a human-computer interface as an add-on -- something which is done when the program is completed to enable people who are not programmers to use the system. What the program actually does is viewed as its substance and the interface is window dressing. This traditional view has its roots in the history of computer science.[2]

Early computer systems used punch cards, then typewriter-like teletypes, and then simple text-only screens as input and output devices. Since one could not make very good interfaces with these technologies, it was natural to view the interface as separate from the program -- an unimportant add-on. Punch cards now exist only in museums, but attitudes about the interface they helped to generate persist.

The process of designing what The Electronic Scrapbook does and how this appears to the user were largely one. I found that it was much easier to make the computer perform a task than to create an interface that would give a non-programmer access to that feature. Interface coding was the most time-consuming part of the task. Furthermore, interface design was often the most challenging part.

As information systems become pervasive and are increasingly used for everyday life tasks, I believe that the design of the interface will become inextricable from the design of functionality.

[1]Current interest in the concept of “agency” makes this definition interesting beyond the author’s intended meaning.

[2]One could also trace this idea back to traditional philosophy which sees rhetoric as mere window-dressing on the true content, or draw analogy to pre-twentieth century conceptions of the mind/body duality: that the body is a mere container for the soul. What these ideas have in common is the conception that outer form is an unimportant container for a true substance which lurks within. Contemporary philosophers would argue that outer form and inner form are and inextricable parts of a whole. A full exploration of this theme is beyond the scope of this thesis.

11.2 The Benefits of Knowledge Representation

I believe that the benefits of semantic knowledge representation increase as the *quantity* of data grows, but decrease as the *variety* of data grows. The first assertion was discussed in section 6.1.4. There is a simple cost/benefit relationship between the time it takes to do knowledge representation and the tasks the information allows one to perform. The more people exist in your video database, the more useful it is to be able to track their family relationships. On the other hand, if one needs to track not only their family tree but also their medical, legal, and financial records and their professional standing, then a semantic approach begins to break down. This assertion is discussed in the following two sections.

11.2.1 Multiple Models

While chapter 6 is called "A Suburban Ontology," it would perhaps be more appropriate to call it "A Suburban Taxonomy" [Sowa 91]. Ontology is defined as "the science or study of being"; whereas, taxonomy is a synonym for "classification" [OED 71]. In creating a computer representation of the world, it would be presumptuous to say that one is getting at the essence of being; rather, one is creating a classification that is appropriate to the task at hand.

An assumption of this thesis is that there are sufficient similarities between families that one taxonomy will be useful to a large number of people. It should be useful in particular to American, late twentieth century, middle-class families -- the type of people who are likely to own camcorders. The validity of this assumption has not yet been tested empirically with user trials.

One ontology will certainly not meet everyone's needs. It is desirable for a video logging system to have multiple models for different places, times, and purposes. At its simplest implementation, one might choose from a library of taxonomies: the east coast family model, the midwest family model, the Quebec family model, the Los Angeles corporate model, the Berlin corporate model, the Louisiana legal model, and the like. Ideally, models should be able to coexist and interact.

11.2.2 The Need for a Common Language

Even if one has multiple models, there is a further problem: it is fallacious to assume that one author's categories will make sense to another. What is one person's "dresser" is another person's "chest of drawers." The task of tracking such synonyms is non-trivial. For example, the 1991 edition of *Medical Subject Heading*s, which attempts to create a common language for medical researchers, is 1,152 pages long and contains 16,131 entries [MeSH 91]. Seven people work full time maintaining MeSH -- three creating new headings and changing headings, one looking for global patterns in the information, one director, one administrator, and one systems support engineer. The 1992 edition of MeSH will contain 16,640 entries, which includes approximately 555 additions and 46 deletions [Schuyler 91]. Creating a common language is no small task.[3]

By having users select from lists, The Electronic Scrapbook reduces the likelihood that the same thing will be called two different names. However, this is decreasingly effective as the length of lists gets large -- from a ten item list, it is unlikely that someone will include both "dresser" and "chest-of-drawers"; from a 1000 item list, a user accustomed to the term "dresser" is unlikely to notice that "chest-of-drawers" is already included. This is not a surface problem, but a fundamental one which draws into question the utility of doing large amounts of knowledge representation.

The Electronic Scrapbook uses a small, special-purpose taxonomy which can be extended by the user. I believe that libraries of taxonomies for specific tasks will be useful. I do not believe that one could create a useful general-purpose ontology. In other words, I am doubtful that this approach will scale. However, it will be useful to accomplish a variety of practical applications.

[3]This standardized medical vocabulary is used to index articles from medical journals. However, individual physicians use their personal notation to write patients' records. The problem of standardizing medical records is analogous to creating a large video database. The personal notation used in creating a video log may not be meaningful to others trying to access that information in a database unless a standard language is agreed upon.

Bibliography

[Andrew 76]	Andrew, J. Dudley. *The Major Film Theories*. USA: Oxford University Press, 1976.
[Barthes 77]	Barthes, Roland. *Image -- Music -- Text*. New York: Hill and Wang, 1977.
[Bloch 87]	Bloch, Giles R. "From Concepts to Film Sequences." Short paper, Yale University Department of Computer Science, Artificial Intelligence Lab, 1987.
[BD 89]	Brondmo, Hans Peter and Glorianna Davenport. "Creating and Viewing the Elastic Charles -- a Hypermedia Journal." Presented at *Hypertext2*, York, England, June 1989.
[Bruner 86]	Bruner, Jerome. *Actual Minds, Possible Worlds*. Cambridge, MA: Harvard University Press, 1986.
[Burch 81]	Burch, Noel. *Theory of Film Practice*. Princeton, NJ: Princeton University Press, 1981.
[Clarke 84]	Clarke, Timothy J. *The Painting of Modern Life: Paris in the Art of Manet and His Followers*. Princeton, NJ: Princeton University Press, 1984.
[Clarke 87]	Clarke, Timothy J. Lecture given at Harvard University in 1987.
[Cook 81]	Cook, David A. *A History of Narrative Film*. New York: W. W. Norton and Company, 1981.
[DSP 91]	Davenport, Glorianna; Thomas G. Aguirre Smith, and Natalio Pincever. "Cinematic Primitives for Multimedia." To be published in a special issue of *IEEE* on multimedia, summer 1991.

[Davenport 87] Davenport, Glorianna. "New Orleans in Transition, 1983-1986: The Interactive Delivery of a Cinematic Case Study." MIT Media Laboratory. Paper presented to the International Congress for Design Planning and Theory, Boston, August, 1987.

[Davis 91] Davis, Marc. "The Director's Workshop: Semantic Video Logging with Intelligent Icons." In *Proceedings of the Intelligent Multimedia Interfaces Workshop*, American Association for Artificial Intelligence conference, July 1991.

[Dehn 81] Dehn, Natalie. "Story Generation after Tale-Spin." In *Proceedings of the Seventh International Conference on Artificial Intelligence.* New Jersey: Los Altos, California: William Kaufmann, 1981.

[Dehn 89] Dehn, Natalie. "Computer Story-Writing: The Role of Reconstructive and Dynamic Memory." Ph.d dissertation, Yale University, 1989.

[Greenberg 61] Greenberg, Clement. *Art and Culture; critical essays.* Boston: Beacon Press, 1961.

[Haase 90] Haase, Kenneth W. Jr. "Arlotje Internals Manual." Internal document, The Media Laboratory, Massachusetts Institute of Technology, 1990.

[Haase 91] Haase, Kenneth W. Jr. Personal communication, 1991.

[HCM 90] Hammond, Kristian, Timothy Converse, and Charles Martin. "Integrating Planning and Acting in a Case-Based Framework." In *Proceedings, Eighth National Conference on Artificial Intelligence*, American Association for Artificial Intelligence (AAAI). Cambridge: The MIT Press, July-August 1990.

[Harel 90] Harel, Idit (ed.). *Constructionist Learning.* Cambridge, MA: MIT Media Laboratory, 1990.

[Hewitt 91] Hewitt, Carl. Lecture at the MIT Artificial Intelligence Laboratory. Cambridge, MA: May 2nd, 1991.

[IRIS 89]	"IRIS Intermedia: Pushing the Boundaries of Hypertext," *The Seybold Report on Publishing Systems* 18:21, August 1989.
[KS 84]	Kolodner, Janet and Robert Simpson Jr. "Experience and Problem Solving: A Framework." In *Proceedings of the Sixth Annual Conference of the Cognitive Science Society*. New Jersey: Lawrence Erlbaum Associates, 1984.
[LG 90]	Lenat, Douglas B. and R. V. Guha. *Building Large Knowledge-Based Systems, Representation and Inference in the Cyc Project*. Reading, MA: Addison-Wesley Publishing Company, 1990.
[Leacock 88]	Leacock, Richard. Lectures given at the Massachusetts Institute of Technology from 1970 to 1988.
[Lehnert 81]	Lehnert, Wendy. "Plot Units and Narrative Summarization," in *Cognitive Science*, 4:(293-331), 1981.
[MD 89]	Mackay, Wendy E. and Glorianna Davenport. "Virtual Video Editing in Interactive Multimedia Applications." *Communications of the ACM* 32:7:802-810, 1989.
[MeSH 91]	*Medical Subject Headings*. National Library of Medicine, US. Department of Health and Human Services. Bethesda, MD, 1991.
[Meehan 81]	Meehan, James. "Tale Spin." In *Inside Computer Understanding*, edited by Roger Schank and Christopher K. Riesbeck. New Jersey: Lawrence Erlbaum Associates, 1981.
[Minsky 85]	Minsky, Marvin. *The Society of Mind*. New York: Simon and Schuster, 1985.
[Nelson 90]	Nelson, Theodor Holm. "The Right Way to Think About Software." In *The Art of Human-Computer Interface Design,* edited by Brenda Laurel. Reading, MA: Addison-Wesley Publishing Company, 1990.

[Nilsen 85] Nilsen, Vladimir. *The Cinema as a Graphic Art*. New York: Garland Publishing, 1985.

[OED 71] *Oxford English Dictionary*. USA: Oxford University Press, 1971.

[OED 87] *Oxford English Dictionary*, *Supplement*. USA: Oxford University Press, 1987.

[Papert 80] Papert, Seymour. *Mindstorms*. New York: Basic Books, 1980.

[PA 84] Pincus, Edward and Stewart Ascher. *The Filmmaker's Handbook*. New York: Penguin Books, 1984.

[Pincever 91] Pincever, Natalio. "If You Could See what I Hear: Editing Assistance Through Cinematic Parsing." Master's thesis, The Media Laboratory, Massachusetts Institute of Technology, June 1991.

[Redmond 90] Redmond, Michael. "Distributed Cases for Case-Based Reasoning; Facilitating Use of Multiple Cases." In *Proceedings, Eighth National Conference on Artificial Intelligence*, American Association for Artificial Intelligence (AAAI), July-August 1990.

[RS 89] Riesbeck, Christopher K. and Roger C. Schank. *Inside Case-Based Reasoning*. Hillsdale, NJ: Lawrence Erlbaum Associates, 1989.

[Rubin 89] Rubin, Benjamin. "Constraint-Based Cinematic Editing." Master's thesis, MIT Media Lab, June 1989.

[RD 89] Rubin, Benjamin and Glorianna Davenport. "Structured Content Modelling for Cinematic Information." Workshop on Video as a Research and Design Tool, MIT, March-April 1989.

[Sasnett 86] Sasnett, Russell Mayo. "Reconfigurable Video." Master's thesis, MIT Media Lab, February 1986.

[SA 77] Schank, Roger and Robert Abelson. *Scripts Plans Goals and Understanding*, New Jersey: Lawrence Erlbaum Associates, 1977.

[Schroeder 87]	Schroeder, Carl. "Computerized Film Directing." Bachelor's thesis, Massachusetts Institute of Technology, May 1987.
[Schuyler 91]	Schuyler, Peri. Head of *Medical Subject Headings*, National Library of Medicine. Personal communication, July 1991.
[Shu 88]	Shu, Nancy C. *Visual Programming*. New York: Van Nostrand Reinhold, 1988.
[Sowa 91]	Sowa, John F., ed. *Principles of Semantic Networks*. San Mateo, CA: Morgan Kaufmann Publishers, 1991.
[Travers 89]	Travers, Michael. "A Visual Representation for Knowledge Representation." In *Hypertext '89*. New York: ACM Press, 1989.
[Winston 84]	Winston, Patrick Henry. *Artificial Intelligence*. Reading, MA: Addison-Wesley Publishing Company, 1984.
[Zimmermann 90]	Zimmermann, Patricia. "Reel Family." Ph.d dissertation, The University of Wisconsin at Madison, 1984.