Reconfigurable Video

by

Russell Mayo Sasnett

Bachelor of Arts
Dartmouth College
Hanover, New Hampshire
1982

SUBMITTED TO THE DEPARTMENT OF ARCHITECTURE
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS OF THE DEGREE

MASTER OF SCIENCE IN VISUAL STUDIES

AT THE MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February, 1986

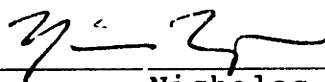Copyright Russell Sasnett, 1985

Signature of the author _____
Russell Sasnett
Department of Architecture
January 17, 1986

Certified by _____
Richard Leacock
Professor of Cinema
Thesis Supervisor

Accepted by _____
Nicholas Negroponte
Chairman
Departmental Committee for Graduate Students

- 1 -

Acknowledgements

First and foremost, I must thank Rosalyn Gerstein for inspiring me with the potential of new media to address social problems, and commend her for the 'persistence of vision' required to bring her project, "Marital Fracture," to fruition over a three-year period. This documentary video portrait of a couple in the process of separation provided fertile ground for us to shape a dream of a new kind of video experience, which we termed 'Reconfigurable Video.' Most of the ideas presented in this paper resulted from long discussions with Roz over how to deliver the best 'exploratory' multi-media resource, and they were first put to the test using her material as an experiment [Gerstein]. I feel fortunate to have had the opportunity to work with such an extraordinary person, whose ideas and efforts have taken us so far. I am also greatly in her debt for all the support and enthusiasm she offered during a difficult time for me. Thank you, Roz.

I would also like to thank Peter Roos, with whom I began the long descent into programming-mode. His insight into the essentials of interactive programming structures provided a foundation for everything that was to follow. Also John Thompson, who donated a homemade database manager that never let me down. And Sarah Griffith, for using my editing system and almost liking it, and for her helpful criticism of my work.

Credit must also go to Benjamin Bergery, for introducing me to the medium of videodisc and intriguing me with its complexity and potential for improvement. The "Elastic Movies" project under his guidance was a profound experience for all involved.

A special thanks to Ricky Leacock and Glorianna Davenport for their votes of confidence, and unwavering support through what was often a meandering course. They helped me follow my interests, when I could find them. Also to Keishi Kandori, who not only wired the Edit Systems Lab, but was always available to help me solve any technical difficulties.

Generic thanks go to all those who cannot be named, for all the little favors and extra efforts that make it possible to get things accomplished, and especially: Jason Kinchen (equipment manager, filmmaker, and pal), Diane Bohl, Vanessa Boris, and Steve Kuettel.

Lastly, thank you to my parents for making it possible for me to come to MIT. They instilled in me a strong sense of the need to find and solve problems, and taught me to take pleasure in doing it.

Reconfigurable Video

by

Russell Mayo Sasnett

Submitted to the Department of Architecture
on January 17, 1986

in partial fulfillment of the requirements
for the Degree of

Master of Science in Visual Studies

ABSTRACT

The problem of efficient access to motion-picture
content is explored, particularly when it is being used
as an information resource. Traditional linear-access
modes do not provide for a personalized presentation,
and most computer-aided video instruction systems do not
allow for exploration.

A solution is proposed wherein a database model
describing the video materials and their connectivity
may be 'published', allowing for content-specific
access, perusal, and viewer modification in the final
installation. Methods for obtaining the descriptive
data are explored, with consideration given to those
which make use of existing data generated during video
production.

A project is described which implements many of
the proposed methods in a personal-computer environment.
The database system and its representational model are
explained, and a set of pictorial access conventions are
proposed that use computer graphics to simulate editing
processes, providing visual access to a visual medium.

Thesis Supervisor: Richard Leacock
Title: Professor of Cinema

TABLE OF CONTENTS

I. Introduction

Reconfigurable Video is a term invented to describe a particular delivery method for video products, whereby subject matter and presentation format may be varied for different audiences using the same video source materials without recourse to programming. The purpose is to increase the versatility of video resources by allowing new content-specific accessing modes to be created for them by viewers. The Reconfigurable Video system is implemented as a library of video imagery and an associated computer database called a VideoFile, which together yield a range of movie experiences as they are perused.

"Reconfigurability" refers not only to the possibility for rapid rearrangement of video segments, but also, in a broader sense, to an ability to restructure the content delivered by the medium. A single library of video materials may be capable of serving a myriad of applications, potentially ones not originally envisioned by the producers. Movie presentations can be constructed which cater to a wide variety of viewer preferences: editors can filter out pathways for display to specialized user groups, and experts can find and scrutinize segments of visual data.

This is accomplished by creating a large central index or repository of information into which any kind of textual data or database descriptor record accompanying a video

project may be 'dumped' -- transcripts, logs, notes, edit lists -- and then establishing associations between this information and the actual video addresses. This cross-referencing aid is called the VideoFile, which acts as both interface and companion to all subsequent users of the video product.

In its later stages (after potential transmission, publication, or installation) the VideoFile is used primarily to group sets of information on similar topics, establishing a network of links between segments. As more people use the VideoFile, they build upon the knowledge others have gleaned from it. Eventually, every topic covered by the video should have several pathways available for browsing, and very little effort on the part of viewers should be required to obtain precisely the subset of information they are looking for. As an example, students shouldn't have to watch a two-hour film to find the ten-minute segment pertaining to their area of interest. This is the kind of use of video, as a resource material, that the concept of Reconfigurable Video particularly seeks to address.

The VideoFile represents a significant contribution to the consumption of video by allowing for the creation of indices and outlines, and active methods of previewing and browsing information which have previously been unavailable to the medium. Just as these operations enable new kinds of

structural interface to the material, so must new modes of physical interface be developed, so that the manipulation and reorganization of segments can be as engaging and intuitive as the original process of film editing. Although Reconfigurable Video is explored primarily as a tool for videodisc distribution, an attempt is made to present viable methods for the establishment of similar systems in the realm of videotape, as used by both professionals and consumers.

Care has been taken to design a system and set of conventions which would be useful in an educational or consumer-level context. The goal was not so much to produce a prototype of such a system, but rather to develop through exploration a foundation of guiding principles which would be generalizable to a wide range of possible applications. Demonstration software was constructed to test the feasibility of many of these precepts, using relatively inexpensive hardware and uncomplicated software techniques; all programming was done in C under MicroSoft DOS, using an IBM PC-XT and AT. The results of these tests helped shape further research. The final step was the integration of many of the tools developed into a videodisc delivery system, which was used to establish a cross-referencing network for a project called "Marital Fracture", a piece which combines affective documentary video with a host of other information and resource materials.

I have explored what benefits might accrue from treating motion pictures as a valuable information resource in and of themselves, and have attempted to formulate designs for computer-aided access, developing demonstration and feasibility software where possible. My particular focus has been on the informational film, and my motivation and desire to make the transmission of content through cinema a more efficient process. This paper combines a presentation of methods with a rationale for their instigation. The second section outlines some problems of video access as commonly experienced by consumers, educators, and professionals, and points the way towards some methods for their solution. The third section delineates a data model for motion picture content, describes some kinds of data which traditionally are associated with motion pictures, and investigates some methods for obtaining the data. The fourth explains how this model may be used to construct new accessing modes for video source materials.

II. Video Accessing Problems

II.A. Problem Definition

Imagine yourself residing in one of the 18,000,000 American households which are part of the exploding Video Generation [Newsweek], complete with cable hookup, VCR (Video Cassette Recorder), and high-resolution television set. Last night you recorded a popular weekly news magazine show while you were having dinner with friends, proud of your new-found freedom from the network's rigid broadcast schedule. You relax into your favorite armchair to watch a segment about a major automotive manufacturer that everyone was talking about at work today. While waiting for the tape to rewind, you wonder what the other segments on this show are about. You consult the TV Guide only to notice it is out of date. By this time the tape has finished rewinding and started playing back; unfortunately, you forgot to reset the VCR's timer after it got unplugged last week, so it once again missed the beginning of your favorite show. Having no idea where the automotive story is in the program, you resort to laboriously scanning through the entire hour of material, watching intently for the two-second blips that indicate titles or commercials, twice stopping for false alarms. Finally you cue up the correct segment, and sit back to watch, only to discover that it is a rerun of a piece you saw last month.

The above scenario is more likely truth than fiction for the majority of video consumers: they still can't get what they want when they want it. This also appears be the case in industrial and educational video installations, which suffer from a lack of flexibility due to the necessity for programming skills to change them. Also, these systems are designed to deliver skills as a commodity; their message is "do what I say, and you will possess knowledge about X." Few interactive video systems are able to provide for a viewer initiated exploration of a body of materials, but instead insist upon a serial delivery of information. Pupils often find that what has been promised as a 'personal' computer/video teacher turns out to be no more capable of answering a question or satisfying a curiosity than the nearest brick wall. They may feel that their curriculum is welded together by video robotics in an assembly-line fashion, a product built for training rather than a resource developed for learning.

Perhaps at the root of this dissatisfaction is a simple cause: video presentations rarely (if ever) divulge their agenda directly to their viewers. There is always a built-in media barrier which prevents direct access, and some mediating agent which leads viewers along a path for an external purpose: directors want to tell a story, reporters want to impress an opinion, programmers want to deliver a course. It is the surrender to this implicit

agent which makes video a passive medium as much as the lack of appropriate technology.

Therefore, one of the primary goals of a reconfigurable video system is the ability to provide a 'table of contents' for a tape or disc before any user has viewed it, and to play back selected portions according to personal interests. Viewers should have the option of direct access to content just as readers do. Print consumers never have to struggle through a volume without a table of contents; why should video users be less privileged? Why should viewers have to 'read the book' to know what it is about? And why should only one mode of access be available, namely, 'cover-to-cover'?

Here a distinction can be made between what is commonly referred to as an Information Resource and what is deemed Entertainment. Popular perception of video is as a home deliverer of Hollywood-inspired entertainment, less than movies and less than art, and certainly not something useful. Just as it is not very important for a novel to have a detailed index or table of contents, yet critical in an academic textbook, so different requirements will be found for the various video genres. Currently the videotape sales industry is experiencing the backwash of the 'how-to' movement, as consumers realize that their VCR's can do more than play movies, teaching them how to get in shape, fix

their plumbing, or even produce a video family album. Recently the first VCR game was introduced based on the popular mystery board game, "Clue." All such non-movie presentations, however, suffer from the inability of current consumer-level equipment to present material in anything but a sequential manner. The other problem is that even if such inexpensive technology were widely available, it would be at best a clumsy process to find the desired segments without an adequate method for publication of frame references.

Somehow, the recording and indexing of the content of video materials in a modern Age of Information has not been thoroughly addressed. Mostly this is due to the ephemeral nature of television broadcast, which dominates the way the medium is popularly perceived. Programming is cast out into the ether, where viewers catch a fleeting glimpse of it before it is gone forever. All of that has changed with the introduction of the consumer Video Cassette Recorder (VCR). With over 30% of U.S. households currently owning VCR's and an additional 1.2 million units sold each year [NewsWeek], it seems likely that every frame of television that is broadcast is being preserved for posterity somewhere. The act of transmission enters these works into a kind of private 'public domain': consumers are free to maintain their own personal video libraries, at least for the time being. [Strong]

The world's body of audiovisual 'literature' is growing
at an alarming pace, with no sign of a corresponding library
filing and reference system emerging. All copyrighted films
must be registered with the Library of Congress, and receive
some classification in this way. But since it is currently
impossible for the general public to address any subportion
of an audio-visual work, information stored about a movie
communicates only the fact of its existence and a vague sense
of its contents in a short abstract. Imagine if no literary
reference could be made to page numbers; a Greek scholar
might refer to Homer's work by saying, "read 'Odyssey';
sometime during the first few hours you will come upon a
verse about the purpose of the mission." This is precisely
the bind in which film/video scholars, critics, and educators
have been placed, unable to make an explicit reference to a
work or quote a passage. The assumption is always that the
audience has had the shared experience of watching the movie
being discussed, just as students of the theatrical and
dramatic arts must all see the same plays. In fact, each
repetition of a cinematic work is referred to by law as a
'performance', indicating the degree to which our culture has
transferred its attitudes about the theater to motion
pictures. This perception seems to preclude the opportunity
for personalization or responsiveness, a misrepresentation
that is only slowly beginning to change as viewers gain more
control over what and how they watch.

II.B. Towards Solutions

The VCR has brought with it the first real advance in television consumption patterns, allowing viewers the freedom to see what the broadcasters provide at times of their own choosing, via a process referred to as 'time-shifting'. Viewers set automatic timers on their VCR's to record programs which are televised at inopportune hours. Bankers might arrange to record a noon-time edition of 'Wall Street Week' so that they may watch it in the evening, or sports fans might record Monday Night Football so that they don't have to stay up all night. Cable-delivered pay television was supposed to usher in the consumer video revolution, supposedly by delivering more 'responsive' programming to its customers than the advertiser-supported broadcast networks. 'Narrow-casting' or audience segmentation, is a promise yet to be fulfilled, as cable quickly turned into a second-run Hollywood and syndication outlet, dominated by the networks still aiming for the widest possible audience. Other new media developments such as videotext and teletext, which use the home television as an electronic newspaper or bulletin board, have failed to replace the print media overnight as predicted. Instead, home-based video playback and recording equipment has firmly gripped the public's imagination.

It is no accident that the the average household's
largest growing investment in technology goes to its media
entertainment and information outlets. The average American
spends an incredible 6.5 hours per day watching television
[Neumann]. Launched as an interesting experiment with
250,000 participants in 1954, television soared into a
cultural phenomenon with 95% penetration in just ten years
[Owen]. Personal computer manufacturers might do well to
ride the coattails of such phenomenal success into the
homestead. New marketing strategies which pose the computer
as a link between Entertainment and Information in these home
media systems may prove successful in answering the general
public's immediate desire for a rationale for owning a home
computer. The computer might be portrayed as the heart of a
burgeoning Home Media Center: vigilant monitor of airwaves,
faithful button-pushing servant, meticulous librarian.

It seems inevitable that the personal computer should be
married to the home VCR, videodisc player, and television to
function in its natural role as information arbiter. As
transmission schedules become more complex and public
awareness of computer-assisted information services
increases, consumers may discover a use for the home computer
in managing their television viewing and in editing and
accessing their personal video libraries. Although
interactive video systems have existed for some time in
professional and educational circles, they remain expensive

due to a confused marketplace and the small economies of scale, and are not 'generic video interfaces' because of the training functions they must perform.

Three technical requirements must be met in order to expand consumer (low-end) computer-aided video functionality. Expensive physical access barriers must be removed, allowing for more direct access to content:

1) computer control of all video playback devices must be standardized

2) addressing data must be embedded in all video materials, transmitted or recorded

3) a flexible format for a descriptive model of video works must be developed, so that meaningful and efficient directories to content may be created, independent of delivery system specifics (hardware or software). Corollaries to The Dewey Decimal System, The Table of Contents, and the Index must be developed for video.

Once these constraints are met, the way will be clear for the popular use of video as an effective resource material in addition to entertainment and training. Video playback devices will become simply computer peripherals, with standardized methods for interfacing, storing and retrieving

video segments, and building directory structures. This will make it much easier to deliver to video consumers "what they want, when they want it."

Imagine a different scenario from the one first presented, given the satisfaction of the above-stated conditions. Returning home from work, you visit the Media Computer which monitors television transmissions for you, and ask for the Video Librarian. It tells you that three programs have been recorded since you last inquired; one of these is a weekly news magazine show you overheard being discussed at work. Opening its file, you are presented with a table of contents outlining the three stories in the program. You instruct the machine to show only the segment about the automotive manufacturer for you, at which point the VCR shuttles to the appropriate section of the tape and starts playing the reporter's opening comments.

II.B.1. Standardized Computer Control of Video Devices

The first step towards stardardized computer control has been taken by the establishment of a SMPTE (Society of Motion Picture and Television Engineers) Subcommitee for the Digital Control of Television Equipment. This group has published several Recommended Practices for the creation of data communication networks in television production facilities, as well as 'dialects' for videotape recorders (VTR's) and

video switchers [SMPTE]. Based on high-speed RS-422 serial communications (maximum 38.4K bits/sec, at up to 4000 feet), these signals are capable of being generated by any computer with inexpensive hardware. The goal of this group is to standardize the ways in which computers control video equipment so that the emphasis in system development will be on software and integration rather than on implementation of basic functionality.

> Once the remote control system is software-based, the broadcasting industry will have been finally able to divorce the question of human ergonomics from the question of the needs of the processing path. This will enable the development of the post-production industries to proceed.
> -- Progress Report on EBU-SMPTE Serial
> Data Control of TV Equipment, Bob McCall
> February 3, 1983 presentation to SMPTE

Until very recently, professional video machines were capable of being controlled only through contact-closure interfaces, which required the use of an Intelligent Interface (I-Squared) to convert simple and inexpensive serial data from a central computer to the parallel voltages necessary to push buttons on the machine. Unfortunately, each machine's interface has generally been unique, requiring much ingenuity (and money) to construct an individual

Intelligent Interface for each and every component in a professional video system. Computer people are always amazed at the prices of computer-controlled video editing systems given their abysmally poor user-interface, lack of programmability, and extremely limited functionality for the most basic kinds of data processing. The main reasons for these deficiencies seem to be threefold: 1) custom interfacing to video devices is expensive and wasteful of resources, 2) the emphasis is on accuracy and machine-control capacity first and foremost, and 3) video people are not familiar with real computer systems, and generally don't know what they are missing. [see Figure 1]

These difficulties have left few resources to develop the human usage aspects of previous systems. Recently newcomers to the professional video editing market have found tenuous niches in supplying these much-needed user-interface features. EditDroid by Lucasfilm's DroidWorks and the Montage Picture Processor both appeal to specialized segments, with new material access modes that promise greater creative freedom and speed, in an industry where time truly is money. [Fig. 2] The Montage system in particular accomplishes all of its editing tasks through the manipulation of digitized sample frames, a process which divorces the editor from the linear-access videotape medium and hence allows a more spontaneous approach. Both of these systems do not actually 'perform edits' in the traditional sense of cutting and
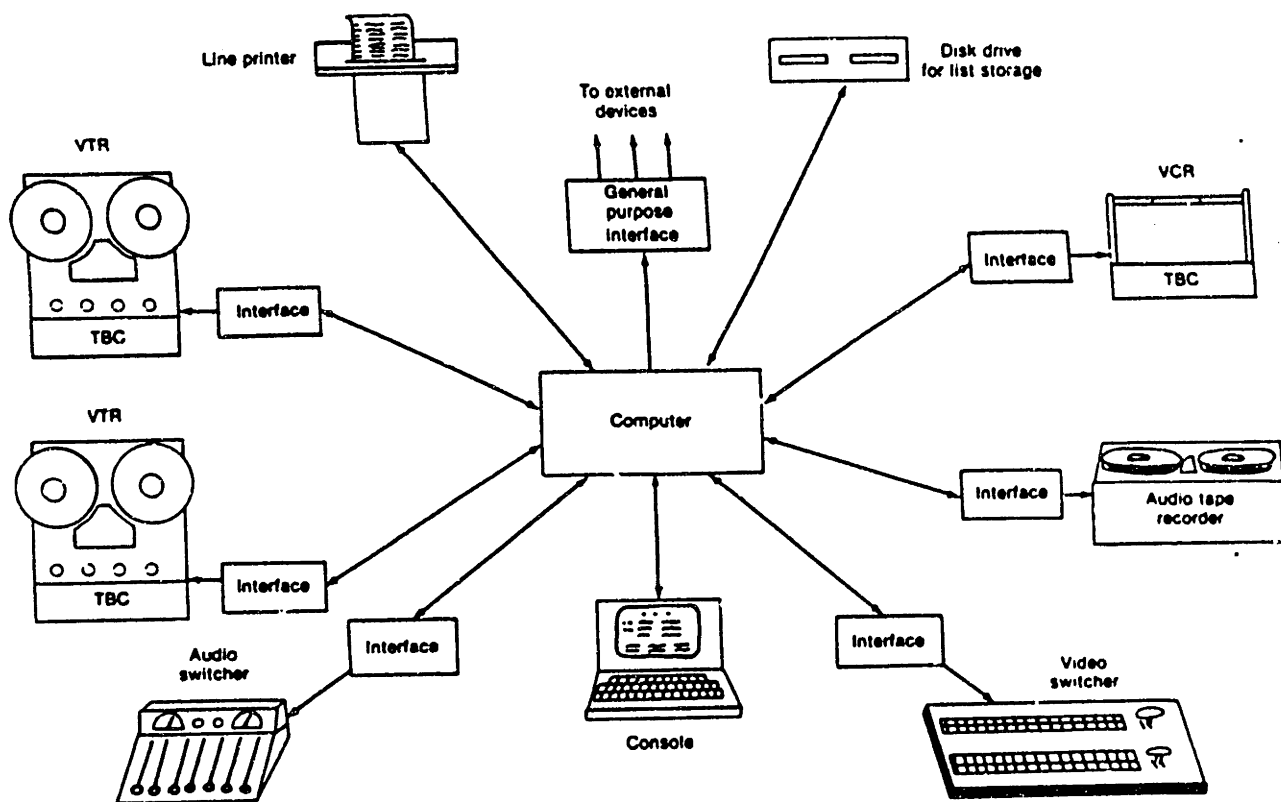
Figure 1. - A typical professional video editing system,
with detail (below) of operator's console
[from Anderson]
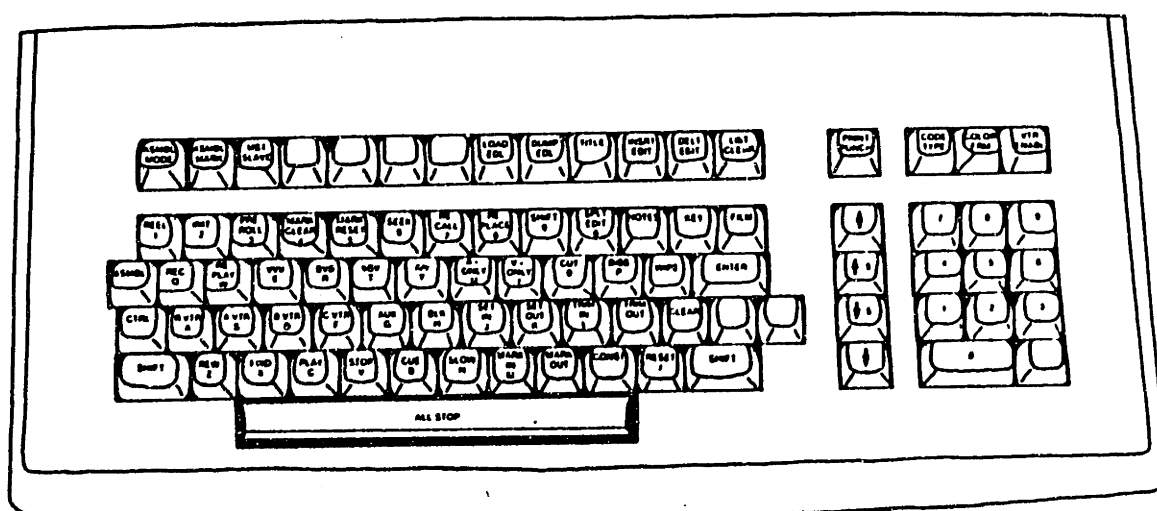
The Electronic Logbook

Logging is the editor's time-honored technique for organizing and controlling raw picture and sound elements of a film or videotape production.

LOG WINDOW
WITH TRIMBIN
FEATURE

EDIT WINDOW
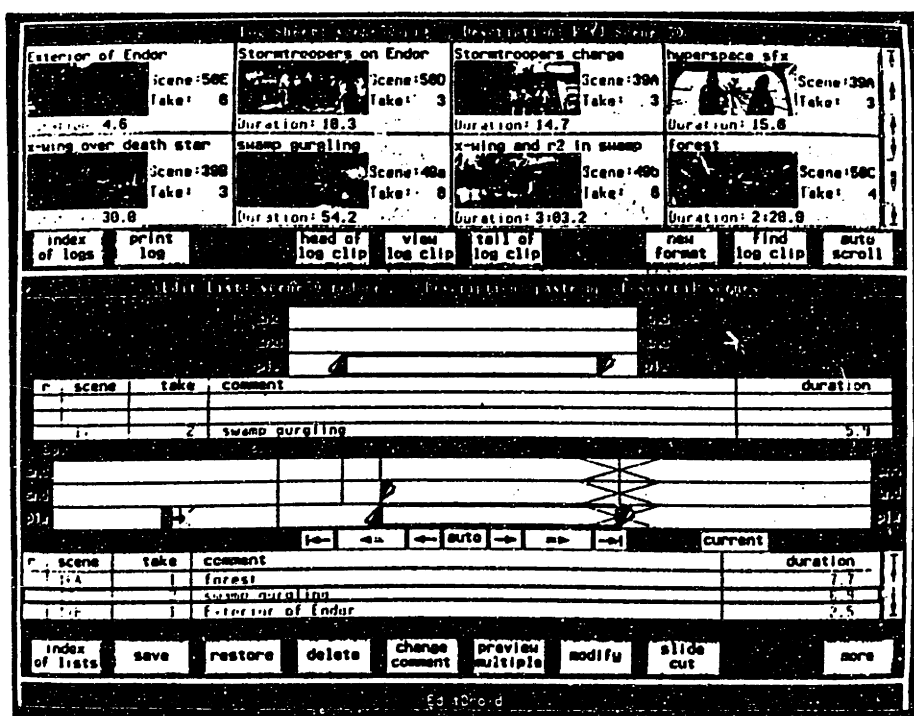
EDIT DECISION LIST

Figure 2.   -   The operator's screen on the EditDroid system
[from Lucasfilm promotional literature]

splicing, but instead offer the possibility of unlimited preview. Unfortunately these systems both use complicated hybrid technologies of videodisc, frame buffers, minicomputers, and proprietary interfaces, pricing them beyond the reach of most small facilities, but their influence upon future system designs is inestimable.

Once the burden of application-specific interfacing is removed, low-cost video system development can turn to such matters as how humans might want to use them. The new SMPTE remote control standard places the bulk of the supervisory burden into distributed processors built into each machine. Versions of this standard have already been implemented in several late models of professional equipment, including the SONY BVU-800 and JVC 8650 VCRs, which has led to the development of several PC-based editing systems with dramatic price reductions.

Soon SMPTE hopes to publish recommendations for remote control of home media equipment, such as 1/2" VCR's and televisions, via personal computer. Perhaps this will engender a rethinking of consumer video needs among manufacturers. Potential benefits to the general public from such a standard are tremendous. At minimum, an interfacing standard based on similar documents prepared for professional equipment would require:

*   a 'physical endpoint connector' description; i.e., 9-pin plug with dimensions and pin assignments

*   a wiring specification, such as twisted pair with individual grounds for noise rejection

*   a data rate, or range of rates, such as 38.4K baud (38,400 bits/second)

*   a protocol outlining message formats and sequences, such as:

    ‹byte count› ‹message ID› ‹parameters› ‹checksum›

*   a 'dialect' for Video Playback Devices (e.g., VCR's and videodisc players) and Video Display Devices (like TVs and Projectors); this allows all of the functions on a unit to be accessed in a similar manner, regardless of manufacturer or format.

    example (for Video Playback Devices):
    ‹58› means STOP
   ‹59› means PLAY
    ‹60› means FAST FORWARD
    ‹61› means REWIND

Agreement on the specifics of communication (input and
output) will allow every entity in a video system to be
treated as a 'black box', making system creation inexpensive
by avoiding needless and wasteful duplication. Witness how
the release of the IBM PC's operating system details helped
create the largest body of available software for any
microcomputer.

II.B.2. Video Addressing Capability

The second problem to solve is the provision of
addressing data, so that every frame of video, whether on a
tape, on a disc, or in the airwaves, may be uniquely
identified. This science has evolved in several stages.
Although the techniques are currently applied only in
professional video post-production systems and in interactive
video installations, the trickle-down effects of
technological diffusion may be expected to occur in time.

Since the earliest days of cinema, it has been possible
to unambiguously identify an exact frame among thousands of
hours of film footage through a process known as "edge
coding." At the time of manufacture, alpha-numeric codes are
exposed onto the edges of the raw stock, between the
perforations, after every 20 frames (in 16mm stock) [Fig. 3].
Thus any frame may be identified by naming the nearest code
and supplying an offset count. These codes are composed of
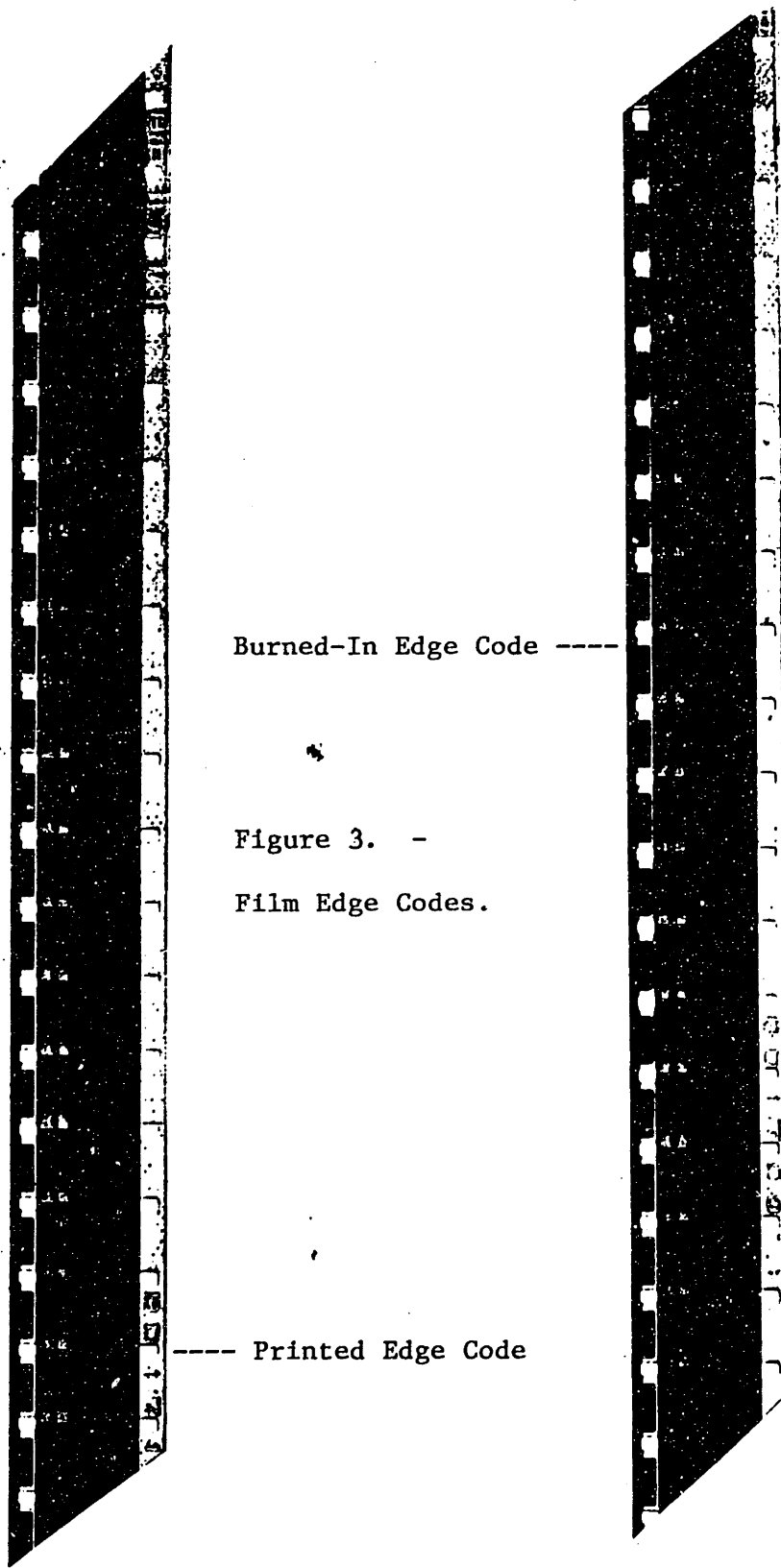
Burned-In Edge Code ----

Figure 3. -

Film Edge Codes.

---- Printed Edge Code

an alphabetic prefix of 1-3 characters and a 7-digit decimal footage count, where each increment represents 2/3 of a second or 6 inches. Therefore the digits alone define a 6.6 million-second address space (1851 hours, or 77 days), and when combined with a 3-letter code at the front represent a possible 3,715 years worth of uniquely identifiable image rendition.

Filmmakers begin the editing process by making work prints from the camera originals, which retain the burned-in edge codes. After months or years of creatively restructuring these work prints, the resulting fine-cut is handed over to a negative cutter, along with the untouched camera originals. It is the negative cutter's task to assemble an exact duplicate of the editor's visual fine-cut by matching the edge codes with those in the original negatives, a tedious and exacting process for which the negative cutter is handsomely paid. The important point to note is that filmmakers never have to concern themselves with these numeric addresses at all (unless they make cuts less than 20 frames in duration, in which case they run the risk of losing the identifying numbers on that image). The process is transparent to the editor/filmmaker because the addressing information is indivisibly embedded in the medium itself. A cut in the imagery makes a simultaneous 'cut' in the addressing data; thus any change, copy, or reconfiguration of the material remains consistent with its

associated data, and any image is ultimately traceable to its original instance. Such a system has only recently been developed for video.

Until about 1967 [EECO], video producers found themselves in much the same boat as modern consumers with their VCR's. The only frame reference possible was obtained by counting Control Track pulses, a 30Hz square wave recorded on videotapes with transitions at the beginning of each frame [Fig. 5]. Unfortunately, this only created a relative address space; it was only possible to refer to a frame as an offset from some other frame, and no frame could be uniquely identified. This made it impossible to search to a frame, or repeat an edit, or make reference to a video segment once the tape had been unthreaded and the reference position lost.

Mostly as an attempt to introduce repeatability to the world of video, SMPTE approved a time-stamp format called Longitudinal Time Code (LTC) in 1969, which gives each frame a time 'address' referenced to a standard 24-hour clock. [EECO] This is accomplished by recording an 80-bit binary code as an analog square-wave pulse train on one of the audio channels of the videotape recorder, in parallel with each video frame [Fig. 4]. At normal play speeds, this is equivalent to reading data at 2400 baud (bits/second) over a serial communications line. But since the data is recorded as a normal audio signal, the time code channel gain must be

VITC

Longitudinal
Time Code

VITC BIT NO.

BIT NO.

SYNC BIT		
SYNC BIT		
UNITS OF FRAMES		
1ST BINARY GROUP		
SYNC BIT		
SYNC BIT		
TENS OF FRAMES		
DROP FRAME FLAG		
COLOR FRAME FLAG		
2ND BINARY GROUP		
SYNC BIT		
SYNC BIT		
UNITS OF SECONDS		
THIRD BINARY GROUP		
SYNC BIT		
SYNC BIT		
TENS OF SECONDS		
FIELD MARK		
FOURTH BINARY GROUP		
SYNC BIT		
SYNC BIT		
UNITS OF MINUTES		
FIFTH BINARY GROUP		
SYNC BIT		
SYNC BIT		
TENS OF MINUTES		
UNASSIGNED ADDRESS BIT		
SIXTH BINARY GROUP		
SYNC BIT		
SYNC BIT		
UNITS OF HOURS		
SEVENTH BINARY GROUP		
SYNC BIT		
SYNC BIT		
TENS OF HOURS		
UNASSIGNED ADDRESS BIT		
UNASSIGNED ADDRESS BIT		
EIGHTH BINARY GROUP		
SYNC BIT		
SYNC BIT		
CRC		
SYNC WORD		

Figure 4. — Time Code Data standards (LTC and VITC).
[from EECO]

- 27 -

Figure 5. - Typical video track assignments.
            [from EECO]

VERTICAL BLANKING INTERVAL
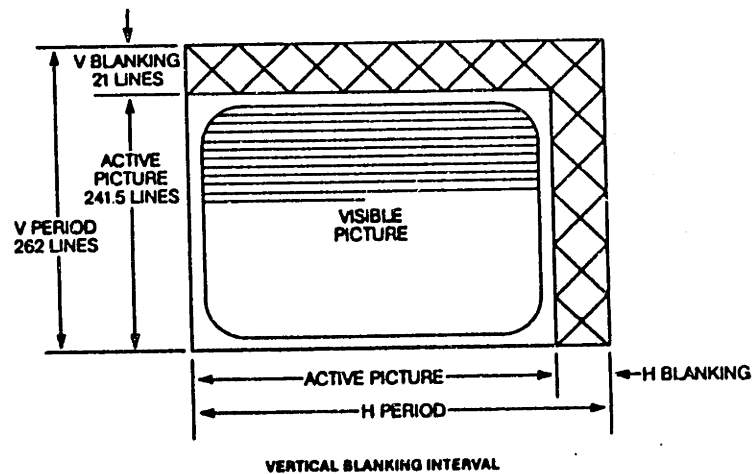
Figure 6. - The Vertical Blanking Interval.
            [from EECO]

adjusted in inverse proportion to tape velocity. Time code is recorded as a saturated high-energy signal, and so must be severely damped at high wind speeds (often up to 30 times normal), and powerfully amplified and filtered at very slow speeds. All of this signal processing requires expensive high-bandwidth audio amplification and filtration circuitry to constantly maintain an accurate time code reading.

Recently, around 1981, a new breed of time code was introduced which is not audio-based. This time-stamp is encoded in the vertical-blanking interval (VBI) of the video signal, where it does not interfere with picture information. This 90-bit word is known as Vertical Interval Time Code (VITC), which provides all of the advantages of the earlier LTC, while also making it possible to read time code when the tape isn't moving or at very high wind speeds. The most interesting thing about VITC (as opposed to LTC) is that the addressing information is truly embedded in the picture channel, just as it is with film edge-coding [Fig. 6]. A variant of VITC is used to encode frame numbers (and other data) on videodiscs [SONY].

Obviously the problem of video addressability has been solved. The real problem is in developing specialized integrated circuits (IC's or 'chips') to perform the tasks of encoding and decoding this data, so that they may be mass-manufactured at favorable costs for inclusion in all levels of consumer gear. The teletext industry has provided

a great impetus for research in this area, since it is their
hope to provide VBI decoders as a built-in option on home
television sets. Unfortunately these data decoders have been
difficult to consolidate into small chips because of all the
noise correction problems that turn them into rather large
analog circuits [Ciciora]. Encoder chips would allow a
time-stamp to be recorded with each frame of video using a
format similar to VITC, identifying the month/day/year and
hour:minute:second:frame of its creation, as well as a short
folio like "Reel 11". The availability of time-stamp reading
and writing could be as valuable to consumers as it is to
professionals; their VCRs could be completely controlled via
computer programs which have searching, editing, indexing,
and cross-referencing capabilities. Imagine what this would
mean to serious video collectors intent on maintaining a
family video history or album, or even just a movie library.

II.B.3. Provision of Descriptive Data

The final problem to solve in order to expand video
accessibility is the provision of descriptive data as part of
the video product, so that it may be split into separately
viewable units which may be selected without the viewer
having to first 'read the book.' This data might best be
provided as an integrated database, since techniques for
browsing and searching such structures are already

well-refined and currently available in all levels of software. The precise content of such a database will be discussed later, but at minimum it must include enough information so these sub-units of the movie may be separately searched out and played back, as well as some textual abstract of their contents. How might such a database be acquired and delivered?

The final output from an editor in video (or film) is often not the finished product, but rather a 'fine-cut' which resembles the ultimate product in all but quality, and directions for the construction of the final release copy. These instructions are then carried out verbatim by highly specialized people (negative cutters) or machines (on-line editing systems). In video, this information is known as an edit list, and it consists of a set of events which must occur at specific tape locations to produce an exact copy of what the editor intends. The edit list is really just a data base which describes the final tape in the most efficient manner possible. Since videotapes are assembled by computers, one could refer to an Edit Decision List (EDL) as Video Assembly Language.

[see Figure 7 for an example of an EDL]

These lists are primarily a communication from one machine to another, and so human comprehension is not an important factor. However, a 'clean' edit list (one without

(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)
001	ISO A	AV	C		01:45:45:00	01:45:50:00	01:00:00:00	01:00:05:00	
002	ISO A	AV	C		01:45:50:00	01:45:50:00	01:00:05:00	01:00:05:00	
002	LINE	AV	D	090	01:45:50:00	01:46:00:00	01:00:05:00	01:00:14:28	
003	LINE	AV	C		01:46:00:00	01:46:00:00	01:00:14:28	01:00:14:28	
003	ISO B	AV	W101	060	01:53:31:00	01:53:46:00	01:00:14:28	01:00:29:28	
004	ISO B	AV	K B		01:53:46:00	01:53:56:00	01:00:29:28	01:00:39:28	
004	TITLE	AV	K	015	00:00:00:00	00:00:05:00	01:00:29:28	01:00:34:28	
005	ISO B	AV	K B	(F)	01:53:56:00	01:54:06:02	01:00:39:28	01:00:49:28	
005	TITLE	AV	K O	015	00:00:10:00	00:00:10:00	01:00:39:28	01:00:39:28	
006	ISO A	V	C		01:46:00:00	01:46:05:00	01:00:49:28	01:00:54:28	
007	ISO B	A2	C		01:54:06:02	01:54:11:02	01:00:49:28	01:00:54:28	

Figure 7. — A typical CMX/ASCII industry standard Edit Decision List.

Field 1 is the event number;
Field 2 is the reel I.D.;
Field 3 is the enabled tracks;
Field 4 is the transition code;
Field 5 is the wipe/key code;
Field 6 is the source in;
Field 7 is the source out;
Field 8 is the record in;
Field 9 is the record out;
Field]0 is the comment.

[from Anderson]

redundancies) is a very useful data set, since it describes exactly, precisely, and completely the locations of all the audio and video segments on the master tape. If these in/out numbers could be married to a larger data set which described the segments in human terms, especially their subject matter or content in some sort of systematic fashion (as in a database), we would effectively have a computerized index of the videotape material. Since an edit list must be constructed to produce almost every master tape, it seems senseless not to make an enhanced version of this data available to the end users. Otherwise, someone will have to recreate the edit list (by logging the materials) in order to reconfigure the video, and none of the information created by the makers passes through to the users.

Perhaps the simplest way to make this information available is by publishing it as part of the finished product. In the case of videodiscs, this is a matter of encoding the data digitally onto the disc itself [Brown]. With tapes, perhaps a vertical-interval data scheme similar to teletext could be used, which would contain the database as part of the video signal. The information would then be retained after dubbing, transmitting, or even re-editing. Just such a data propagation scheme was utilized by NHK in the development of an off-line editing system which retained video addressing data (encoded as VITC) throughout several generations of editing, and then automatically generated the

final master tape by reading back the edit list data from the fine-cut [Kawaguchi].

There are around 10 lines (rasters) of data available in each frame in the vertical interval (lines 10 through 20); VITC encodes 80 bits on two redundant lines (plus 10 bits for Cyclic Redundancy error-checking) [EECO]. Using the same conservative standard, there is room for some 40 bytes 'in-between' each frame (40 characters), for an instantaneous data rate of 1200 characters per second (12K baud). Full use of this channel would allow a one-hour instructional videotape to contain 4.32 Megabytes of data, the equivalent of almost 2500 double-spaced typewritten pages of text.

This is more than enough bandwidth to carry a sizeable database, even for very short programs. The two difficulties are that 1) several other data sources already lay claim to this area of the video signal, such as VITC, closed-caption data, and teletext; and 2) inexpensive VBI data decoders must be built and distributed on a mass scale, as described in the previous section. Also, to realize a unified scheme for data transmission in the vertical interval would most likely require several years of standardization committee proceedings. One possible option for broadcast video is to use the "professional subchannel" of the stereo audio standard proposed by the BTSC (Broadcast Television Standards Committee) which is specifically allocated and designed to carry data

[Danna]. Vertical interval encoding would allow a computer to 'digest' the contents of recorded or transmitted videotapes and create directories for them at the installation site.

II.C. Goals for the Design of a Reconfigurable Video System

To create a system for enhanced video consumption at a general or consumer level, relying heavily upon the technical requirements stated above, the following goals were established:

1)   the system must be content-independent, capable of delivering any movie material once it is formatted to system specifications.

2)   it should be possible to present a table of contents or index for tapes which have never been viewed; this implies a publication of descriptive data rather than a reliance on the users to create it or the system to derive it.

3)   the system should be format-independent; for example, what works for videodiscs should work for videotapes.

4)   the primary orientation is toward novice users.

5) computer graphics should be used to visually represent the physical processes of editing the video material and controlling the video playback device; this is intended to avoid hardware implementations which rely upon banks of special-function keys, and to provide a conceptually consistent method of physical access.

6) viewers can become 'programmers' or editors, capable of altering presentations for future users without a requisite knowledge of programming techniques.

7) a simple procedure should be provided for attaching virtually any type of descriptive data to video segments.

8) viewers should be able to search for specific content without having to watch the video.

9) the system should present coherent mini-movies in response to broad topical inquiries; requests for sufficiently large subsets of the information base should be answered with a movie experience.

10) none of the above capabilities should interfere with standard methods of production, distribution, or consumption.

11) inexpensive or readily available components and
techniques should be used.

III.   A Data Model for Motion Picture Content

Briefly stated, I believe the problem of providing computer-readable movies is essentially a data preservation problem; that is, obtaining data during the appropriate production phases and passing it through to the delivery phases. A good example is the edit list for a videodisc project, which could easily be converted to frame numbers from time code numbers for publication as a ready-made index to the disc. All kinds of gadgets, hardware, and techniques exist for the transmission of data through video channels, including Vertical Blanking Interval (VBI) codes and high-density analog optical storage. Unfortunately, this data rarely has anything whatsoever to do with the video, which merely acts as a carrier. So although techniques readily exist for shipping about movie data, they are not used for primarily economic reasons - greater profit is to be had transmitting news and advertising, and few can imagine just how movie data might be valuable. Also, these technologies are still very young; it was not so long ago that transcript 'data' for the hearing impaired was transmitted by a video insert of a sign-language interpreter instead of by closed-caption VBI codes, and that projects were undertaken to place encyclopedias on videodisc by 'photographing' 54,000 still frames instead of using analog data encryption techniques. So although a delivery method is uncertain, I have proceeded on the assumption that it will be

possible to provide this movie data with each copy of the video product (as stated under the technical assumptions).

A distinction may be made between data normally associated with the production of movies, and that delivered as part of the final product. Two things distinguish these data sets: their users, and the form of their reference to the video product. Suppose we call the first group the Makers, and the second the Viewers. Most of the data generated and used by the Makers could be termed Descriptive; they are interested in keeping track of Things, like production crews, money, and deadlines. It would also help the Makers to know where Things exist in their video tapes, like the best takes and the most interesting dialogue and the blooper that must be cut. In contrast, the Viewers are most interested in keeping track of Ideas, and in finding the answers to questions about Ideas: will this product meet their needs, and where are the most relevant parts, and how might they be used in conjunction with similar materials? This will be termed Associative data.

To generalize, one could say that the task of the Makers is to produce a set of video materials (movie segments) which will meet the needs of some set of Viewers. The task of the Viewers is to find a set of segments which satisfy a curiosity about an Idea. Suppose, then, that the Makers published a Descriptive data model of the video they

produced; and that the Viewers could then browse through an Associative data model which links the segments into movie presentations. It should then be possible for the Viewers to discover matches between what Makers have produced and what they want to see, without having to sift through hours of potentially irrelevant movies. In effect, this is what a Reconfigurable Video System will accomplish.

There are three families of data which will be related to video materials: Addressing Data, Descriptive Data, and Associative Data [see Figure 8].

Addressing data is simply a frame reference, which allows a video image or segment to be retrieved from recorded media, and therefore is necessary for its playback. As stated in an earlier section of this paper, addressing data is assumed to be embedded in the video channel on any future Reconfigurable Video system. Software was developed using both videodiscs and one-inch videotape machines with time-code capabilities to simulate this condition.

Descriptive data defines each movie segment by storing the addressing data necessary for its playback (independent of all other segments) and a host of other data useful in searching fornn its content, such as: short abstracts, thematic categories, visual descriptors, and transcripts. This record-oriented data is stored in a popular relational database format (Ashton-Tate's dBASE III), making it

accessible to a wide range of personal computer software.
This data is to be provided by the filmmakers during the
post-production phase of their work (when it is generated)
and then published as part of the video.

Associative data is used to create links (variable
attachments) between the segment records and other
descriptive data, a network which is created by users for
their particular application. Links of several types can be
made between video and audio segments, blocks of text, and
other database records. Links may be used to create new
edited versions, voice and text overlays, glossaries, or
outlines: almost anything. This data is stored in a custom
network-model database which is generated on the user's
computer.

III.A. Movie Primitives

What are the primitive elements out of which movie
experiences may be assembled?

A thoughtful examination (and some practical experience)
reveals that there are three primitive types of data objects
from which movies (interactive or linear) are built: frames,
segments, and branches (frames, shots, and sequences to
filmmakers). I speak here of movies which are photographed
chemically or electronically, and so avoid in-the-frame
objects composed of pixels which are more the realm of

computer graphics. The frame is the atomic, or smallest accessible unit, and needs no explanation. The segment is a range of frames which lie in one contiguous block on a recorded medium; it consists (minimally) of the beginning frame address and a count of the number of frames in the segment. Since a frame can be represented by a segment of length one, we can condense further to just two data objects -- segment and branch. The branch is an ordered list of segments to be played, and a set of rules for determining the next branch to visit. [Fig. 9]

A linear movie is then a long list of segments to be played, or one large branch. An interactive movie is a network of branches allowing for variable experiences; its level of complexity is related both to the number of branches in the network and the type of branching allowed. Note that (ideally) this model retains the 'program' or instructions for playback in the actual data structures, leading to more generalizable and modifiable programs. There is a rough correlation between this 'movie primitives' model and the more sophisticated model described above: frame = address, segment = description, and branch = association.

This model was explored in a series of programs developed by myself and Peter Roos in the spring of 1984 to implement the flowcharts for the MIT Film/Video production "Elastic Movies", the first computer-controlled (Level III)

Segment Record

```
name: the segment
in: 32768
out  44018
times: 1
speed. 1.0
windows: (list ptr)
comment. good stuff
```

Videodisc

In   **Out**

Segment

1   2   3
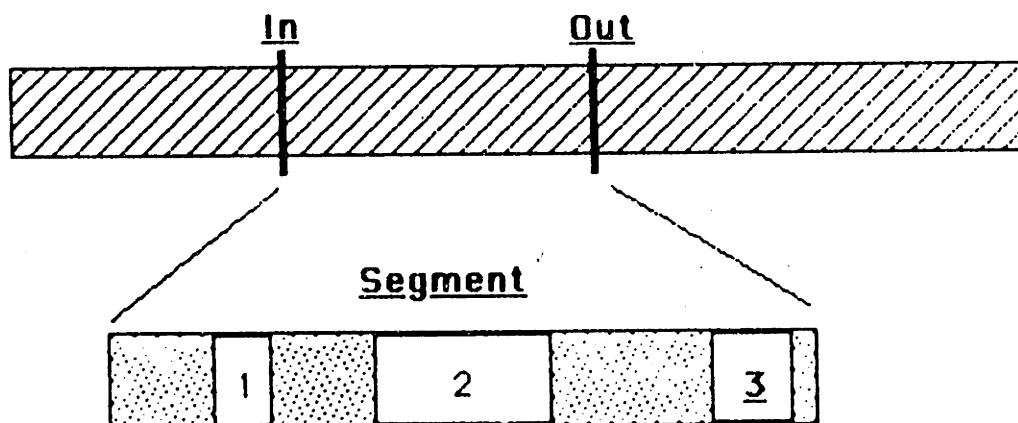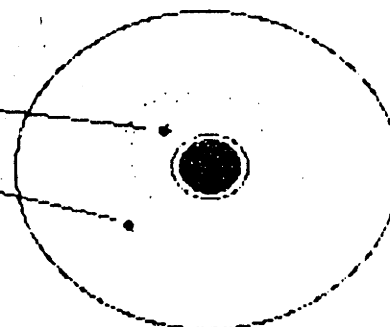
(showing windows 1, 2, and 3)

Figure 9. - Segments.

interactive videodisc by artists. "Elastic Movies" contained
four interactive movie projects: "Dance Haiku", a catalogue
of electronically interpreted dance movements set to music;
"Thirst", a conceptual piece about escape and perseverance;
"Marital Fracture", an experiment in interactive documentary
about a marital breakup; and "Picture Piano", in which story
elements arranged along musical scales invite viewer
improvisation. In addition to working on "Dance Haiku",
Peter Roos and myself volunteered to write software to
execute all four projects.

The result is a keyboard-driven program called CREATE,
which enables the user to interactively shuttle the videodisc
player to desired locations and to define and replay video
segments without using frame numbers. Segment attributes
include: a unique name; the in and out frame numbers; the
number of times to repeat the segment; the playback speed; a
free-text description; and a list of sub-segments called
'windows' which may be used for time-bound interactions.
These segments are stored in hash tables and data files using
the unique user-provided name as the key. Once all segments
for a piece have been defined, the interactive video
experience is defined by creating branch structures. Each
branch contains a list of pointers to the segments to play,
and a set of pointers to new branches to visit based on user
input (via one key-press). The branches are then stored in a
separate data file, also keyed by name. This was done so

that one master set of segments for an entire videodisc (the videodisc segment log) could be used as a foundation for several different interactive movies (branch data files). The mode of interaction may be changed by modifying the routine which maps user input into a choice for the next branch. This was done by including a number of specialized 'control switches' in each branch record which call the appropriate compiled functions on execution.

The display of a particular interactive movie is accomplished by a very simple program which reads in the segments, then reads in the branches and connects them to the appropriate segments, and finally starts the movie given the first, or root, branch. The interactive movie advances automatically by "walking" through the network of branches until it reaches a dead end (a null pointer). [Figure 10] As an example, "Thirst" consists of one long traveling shot following someone down an empty corridor, interrupted at intervals by dreamy dissolves to an other-worldy paradise (described as "a classic dilemma between escape and perseverance"). According to the flowchart, any interaction during the dissolves would take the viewer to that 'other place' for a brief respite. Here the corridor shot is defined as one branch containing one segment, and the dissolves are defined as windows within the corridor segment which trigger (go to) branches about the 'other place.' [diagram] Those branches return to the

Branch Structure

```
name: the branch
segment list: (list ptr)
options: (table ptr)
controls: (switches)
```

Segment List

```
A
B
C
```

Controls

```
interrupts: ON
menu mode: ON
escape: OFF
```

Options

On Input:	Go To Branch:
#17	the next
#42	another branch
#06	still another
#12	you get the idea

Branch Network
(an interactive movie)
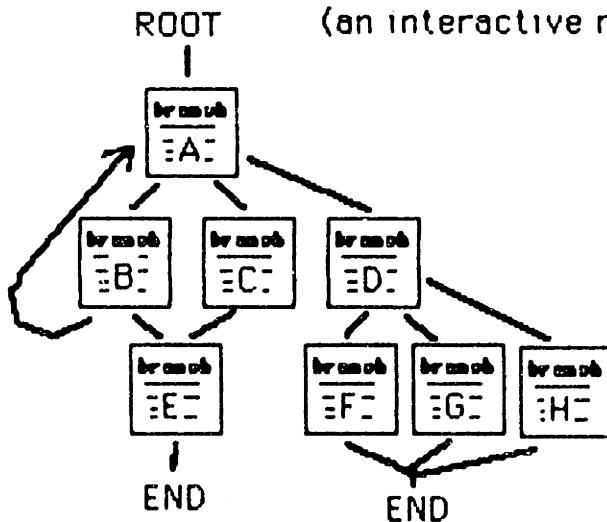
ROOT

Sample Branches

```
'A' options: B,C,D
'B' options: E,A
'C' options: E
'E' options: (none)


Root Branch is 'A'
```

END

END

Figure 10. - Branches.

corridor when they are finished executing. At the end of the corridor, the movie player will find that there are no more branches to visit, and so it returns the user to command level.

Several crucial lessons about interactive video were garnered from this experience. First, the positive discoveries. The main advantage of our approach is that its reliance on the database to encompass the 'program' guarantees that no knowledge of programming or computers is necessary to construct an interactive movie. Also, we were pleasantly suprised to find that a fair amount of generality is insured by making every interactive movie situation capable of being described by just one data structure, the branch.

Negative experiences were numerous. The lack of any coherent overview or map of the command syntax made the authoring process incoherent to anyone except the program's designers; it was thought that a graphical interface would help to narrow input choices. Also, we had ignored the need for providing instructions or a table of contents for each movie, and so it became necessary for the video itself to perform these functions. Finally, we found ourselves up against the Naming Problem; names chosen arbitrarily by one viewer will have little or no meaning to other viewers. These limitations made it clear that this kind of simple data model (segments and branches) is more adept at creating a

presentation system (a Projector), than an exploration system (an Indexor). Really what had been accomplished was to store complicated physical access patterns for playback; the content of each segment remained a complete mystery until it was played.

The next step attempted was to write a graphical interface front-end for the data manipulation routines. This program (VIEW) creates a series of menu screens with a set of graphical Input Devices on each one, allowing a process to be divided into screen-sized chunks of the command vocabulary. Input devices implemented include buttons and double-buttons for function activation, sliders for value input, and scroll-boxes for item selection. The scroll-boxes contain the names of segments or branches, which may be sorted either alphabetically, by location on the disc, or by duration. A digital television was used for display, so that the material being manipulated could be viewed on the same screen as the the graphical input devices which control its playback. [Fig. 11,12] Menu screens are switched to provide new contexts, depending on whether the user is defining segments, creating branches, or playing movies. The resulting system greatly simplifies the input of, and the search for, video segments. However, the definition of branches lacks any visual analogy, and so remains a text-bound and confusing process (except to the programmer).
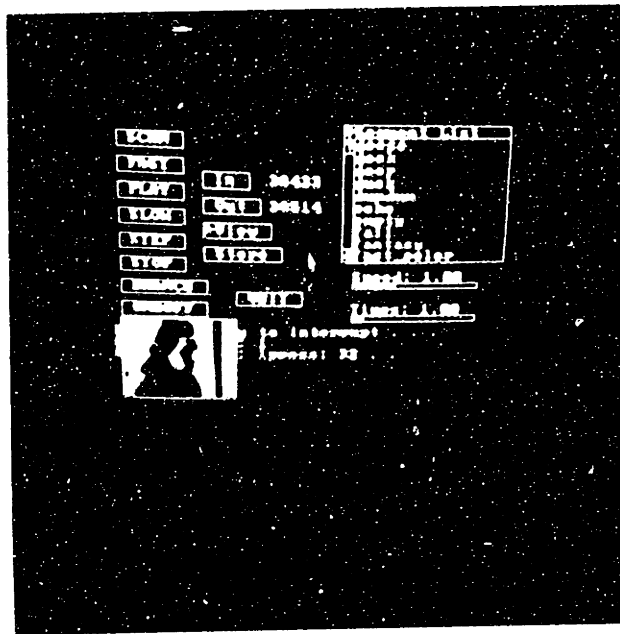
Figure 11. — The VIEW screen. Notice live
video image at lower left.
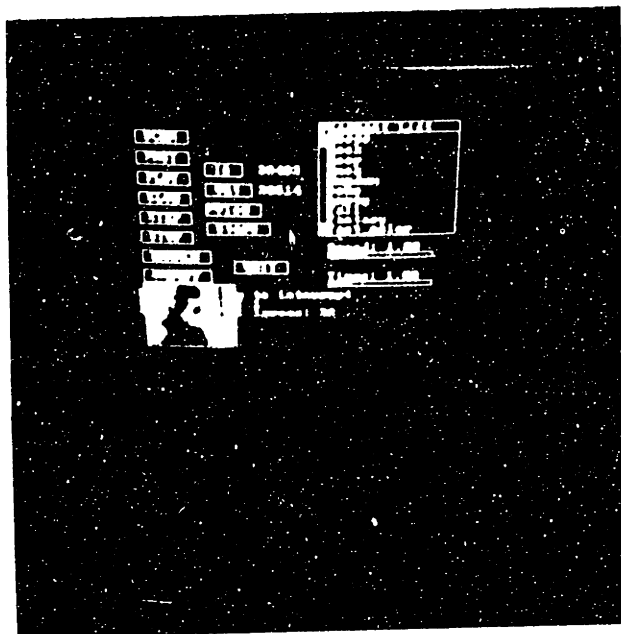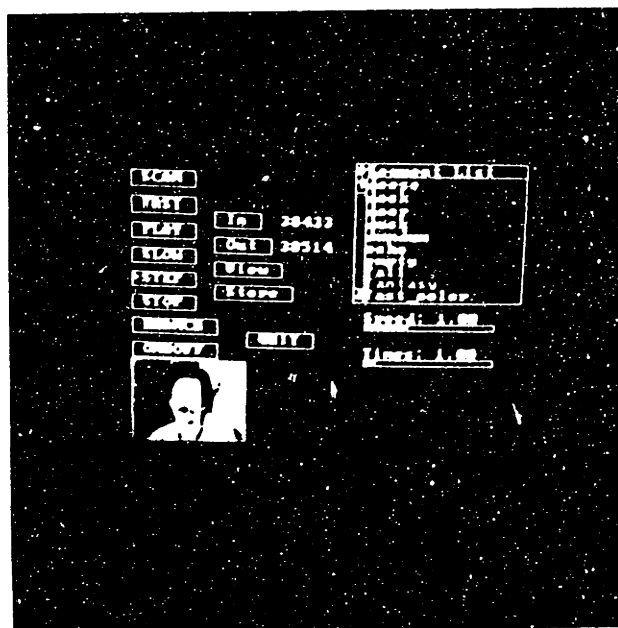
Figure 12. - More of VIEW.
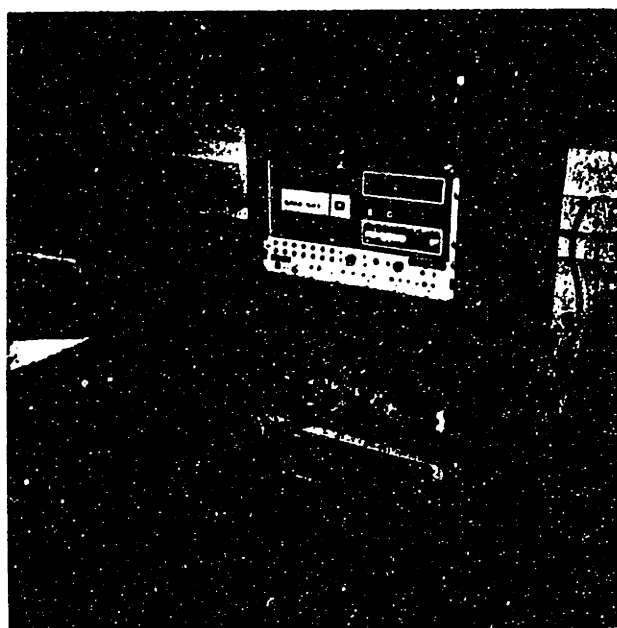
(above - the equipment rack. includes audio amplifier, serial
control CMX audio-follow-video vertical interval switcher, RGB
encoder, genlocking sync generator, serial and audio patch bay,
video patch bay; two SONY LDP-1000 videodisc players nearby.)

At this point it was decided to apply these physical access concepts to videotape as well as videodisc, to demonstrate a measure of device-independence where video playback is concerned. A software interface to an Ampex VPR-2B one-inch VTR was implemented, using a custom-built Z-80 microprocessor-based intelligent interface designed by John Thompson, a graduate of the MIT Computer and Information Sciences program. The remote controller is capable of performing command lists downloaded over a standard serial interface, which are triggered by the timeline generated by the time-code track from a one-inch tape. Several functions may be enabled or disabled at precise tape locations, and time-code reading and searching functions are also provided locally. This interface was used to allow the definition and storage of segments on a tape, and their playback in any order with a minimum of search time. A computer-controlled audio/video switcher with a black channel was incorporated into the playback system, to spare viewers from having to endure the garbled audio and video normally associated with high-speed tape searches. This was affectionately referred to as the "Dump the Junk" feature, wherein videotape viewers would never again have to watch unnecessary portions of tapes. Once the 'important' segments have been defined, the tape will automatically skip over the discarded sections on subsequent playings.

III.B. The Database Format

"The purpose of a database system may be broadly expressed as that of bridging the gap between information and data -- the data stored in memory must be converted to usable information.

A rough classification of information usually includes the following:

Entities

(objects)

Relationships

(between entities)

Properties

(describing entities and relationships)

Actions

(changing relationships and properties)"

-- Alfred Handy Guide: "Understanding Database Management Systems"

A database system must translate communications which are comfortable and natural for its users into a set of actions to be performed on abstract data objects stored in 'files.' A file on a computer is just a block of data; programs are capable of translating this data into different

formats which people interpret as information. Thus a file
may contain a letter to a friend, a picture of a friend, or a
data record which contains a friend's address and phone
number.

Just as you must know your friend's address to send
him/her a letter, so must you know an object's address within
a file to operate upon it. File addresses consist of a file
handle (obtained via the file name) and a byte offset from
the beginning of the file. The delineation of a file
segment, known as a 'block' of data, is accomplished by the
definition of three variables: the file name, the beginning
address (byte offset), and the length of the block in bytes.
[Fig. 13] This is analogous to the situation in video
editing, where a picture segment is defined by a unique reel
name or number (its handle), the punch-in point (beginning
address), and the length of the shot in frames (punch-out
minus punch-in).

III.B.1. The Descriptive Database

Blocks of data may themselves contain sub-blocks. This
hierarchy is often referred to as a 'block structure':

BLOCK STRUCTURE

Files ---> Records ---> Fields ---> Values
        (contain)   (contain)   (contain)

Data File

```
43hjkdsafBy92hrk2ffh
xz      f98986zf32
89      78r037fjd93
8fu     FHaHfRjouyu9
89u90ah 3b  yu  8byd
hfds 9sarOyOHO HHOE
uenfjY(){O)EH}()nother
u09unao9ufg9 bUDUJje
YOYUY (8  bO9uD ybp
79bespuofds     WH jk
(){}{1]\\p\ds
upy&Y&( * *(
798%$yb9yOb- iobsdf
Y(Y *(bpopo()U))(U()Up
```

IN

OUT

A Data Block

Records

Record #1
Record #2
Record #3
Record #4

"Blocks" contain Records

Fields

field the first
name
size
time
(etc.)

Values

"I am a Value"
"and this is my name"
16564
08:47:32

"Fields" have Values

A "Record" contains Fields

Figure 13. — Blocks and sub-blocks of data.

The most general case is served by allowing each record, the largest data block, to be of any size. Most database systems, however, utilize fixed-length records to speed access time, particularly on microcomputers. This is the format used to implement Ashton-Tate' dBASE III, the relational database system chosen for the VideoFile project. dBASE III is used to store (and eventually deliver) descriptions of video segments, which make up the initial core of the database. In the relational model, each file contains a number of records, and each record contains a set of attributes called 'fields'; and every record possesses exactly the same fields. In dBASE III, the fields can be of several different types, including Numeric, Character, Date, Logical, and Memo. Numeric fields represent decimal or floating-point numbers (up to 255 digits), Character fields hold short text strings (up to 255 characters), Logical fields contain truth values (True or False), and Memo fields hold long text segments (up to 4096 characters). Each field has a unique name and a reserved location in the record [Ashton-Tate].

Having each record contain exactly the same fields makes for simplified sorting, viewing, and searching operations, because there is a one-to-one correspondence between an object being described and a record which completely describes it; all actions which operate on the record require no knowledge of other records. Therefore, each record

remains completely independent of every other record (as far as the database manager is concerned). No restrictions are placed on what fields names may be assigned, or the values they may take on, although the information necessary for playing back the individual video segments must be found somewhere in each record. At minimum this will consist of an in-frame and an out-frame, but most likely other information will be included, such as the audio and video channels enabled, and the playback speed. The purpose of each record is to concisely convey the content of its video segment so that it needn't always be viewed when browsing through the database, and to hold the vital statistics for video playback should it be requested.

A number of interface routines to dBASE III were developed using a package from LifeBoat Associates called "dBC III", which provides record-level access to dBASE III files from C. Normally in dBC, fields are accessed via a complicated process of extracting the data from the record block using an offset value and field width, and then converting the ASCII-coded byte strings into the requested type. Often these difficulties force the hard-coding of the offset and width values into application programs, a truly horrifying prospect when integrating large systems. The first set of utilities, written with the aid of MIT undergraduate Earl Yen, are designed to provide simple yet powerful generic field-level access to dBASE III records

using only field names. The process is intiated by calling

DBMKTREE immediately after a new dBASE file is opened; this

creates a binary tree of structures containing information

about each field in the records of that file, ordered by the

unique field names. Each structure contains the name of the

field, its offset into the record block, the width of the

field in bytes, and a type code indicating the kind of data

it contains (Numeric, Character, Date, Logical, or Memo). To

perform a transfer of data, the programmer calls GET FIELD or

PUT FIELD with the name of the desired field and a pointer to

a union which is of sufficient size to hold any of the

values. These routines fetch the field statistics from the

binary tree, and then read/write the field block from/to the

given record block.

The next task was to build browsing and searching

routines for dBASE records. DBROWSE is called with a pointer

to the database and an initial record number at which to

begin browsing; subsequent calls re-enter at the last

accessed record. The initial record is displayed on the

screen, with field names highlighted on the left and values

columnated on the right in a standard "3x5 card" type of data

view. Browsing control is given to the cursor keys 'PgDn'

(forward one record), 'PgUp' (backward one record), 'Home'

(go to a record number), and 'End' (select the record being

viewed). This routine uses low-level IBM system calls for

rapid screen display.

PATTM is the pattern-matching module, not yet completed.
It is currently only capable of searching for patterns in
character-string fields, where:

?   - matches any character

s* - matches zero or more occurrences of character "s"

s+ - matches one or more occurrences of character "s"

-- Functional Description Manual,
Lattice C Compiler

PATTM is given a pointer to a database; it requests a set of
strings to match with the character field values. PATTM
jumps to the first matching record and displays it using the
same routine as DBROWSE. The search can be halted at the
current record or resumed as desired. The goal was to
provide a searching capacity similar to IBM's
Query-By-Example system, wherein ranges are given for numeric
values (less than, same, greater than, or between) and string
patterns for character values in the record structure; the
result of the search is a list of matched records. This
module would be used to search for video materials described
by segment records without having to linearly browse the
database for them.

The last module constructed was VDBPLAY, which plays
back the video segment described by a record. VDBPLAY is

initialized by telling it whether it is attached to a videodisc player or to the one-inch VTR, whether the database uses timecode or frame-number timestamps, and the names of the fields which contain the in and the out points. To play a segment, VDBPLAY is passed a pointer to the database file, and the number of the record to play. Feedback is provided if the source is on tape and a long search must be performed. A video playback may be aborted while in progress by pressing any key.

III.B.2. The Associative Database

The Reconfigurable Video installation system must provide for the retention of relationships between entities, something that the relational model is not very good at, regardless of its name. In theory, the 'relations' in a relational model database are dynamic and temporary, selecting a group of fields or attributes as a partial 'view' of the database, by which data values are subsequently accessed. One database model primarily concerned with the storage of associative relationships is the Network, or DBTG, model [Atre]. This is known as a file-level model because its implementation relies on a set of well-defined data-file operations which link blocks of data. A block of any size and type of data may refer to any other block by establishing a Link or Association of a certain type; this allows for the

creation of sets, lists, and hierarchies of abstract data objects. For example, a video record may be made part of a movie sequence by attaching a link at the tail of the current list of segments. In this case, a link of type "NEXT" is made at the current segment, which points to the new video segment block; the new segment block is given a link of type "PREVIOUS", which points to the current segment [Fig. 14].

The VideoFile network database is based upon a core of routines written by John Thompson. Originally coded for small, efficient applications, the functions were modified to work in the large unprotected- memory model available on the 8086/8088 processor. There are three related fundamental modules from which all application routines are constructed. The heart of the database manager is call "FA" for File Array; its routines allow a disk file to be treated as one large virtual-memory buffer. Blocks of any size may be allocated and freed, and swapped in and out of data structures in core memory. A sophisticated cache- buffering process is used for the low-level file accesses to enhance performance. File pointers to the created objects are returned at the time of their creation. A mechanism is also provided for determining proper closure of the data file, in case a sudden crash caused some cache buffers not to be written, potentially corrupting some of the data. Note that structures and values in FA files are permanent, until specifically destroyed.

Making a Movie by Linking Segment Records
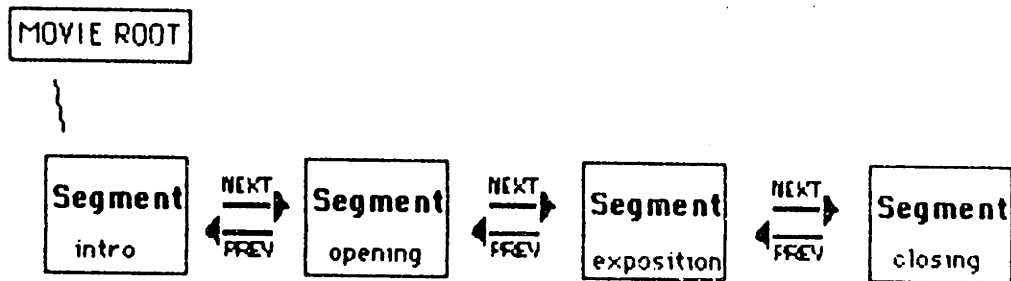
Figure 14. — Making a movie by attaching Links.

"DC" stands for Dictionary, a module whose routines allow for the creation and maintenance of an arbitrarily large 'dictionary' of records. These routines are used when the primary "key" to a database is a word; for example, a glossary of definitions, or a list of word occurences in a file. The DC module automatically handles addition, deletion, and searching of words stored in an FA file as a "2-3 Tree". The 2-3 Tree is a balanced binary tree which is constantly re-adjusted to minimize search time. DC really only operates on the key words themselves; data blocks of any size or format may be attached to the keywords by using the FA routines to allocate extra space to store the keyword and its associated record.

The "AT" module manipulates Attribute Lists, which may be attached to any structure in an FA file via an FA 'file pointer.' The attribute list is an arbitrarily long list of variables and their values, stored as strings of characters. For example, you could create an attribute called COLOR and set its value to "blue." If the FA file contained a number of structures representing objects, you could scan the attribute list of each object to see if it had a COLOR; if you found one that was "blue," you could turn it to "red." AT maintains a DC dictionary based on the attribute types defined in each FA file; this allows attributes to be fetched or written according to their unique type code, thereby avoiding needless and time-consuming string comparisons.

The VideoFile network database was intended to be based
upon the principles of 'object-oriented programming',
particularly as described by Steve Gano in his implementation
of the Movie Manuals project developed at the Architecture
Machine Group [Gano]. Applied to data files, the reasoning
goes something like the following. An arbitrary object may
be represented as a block of data in a file. To interpret
the data, these blocks are given 'type' codes; operations on
the objects then vary according to the kind of object stored
in the block. The difference in interpretation is derived by
the ability to overlay a variety of record structures onto
the data values, giving them different meanings according to
their context. For example, suppose object 'A' contains a
video segment and object 'B' a text segment. A request to
'display A' is handled by determining the type of object 'A'
(video in this case), and then invoking the appropriate
procedure (a video playback routine). Similarly, a request
to 'display B' is also unambiguous, and generates the
expected response. Time did not permit a full
implementation, but the original goal was to maintain DC
dictionaries of the names of each kind of object created.
For example, there would be a dictionary of type MOVIE
containing an alphabetized list of movies created so far.
Such a Data Dictionary would make for a self-documenting
installation, a particularly attractive situation when the
users are novices (i.e., not programmers).

III.C. Obtaining the Data

Given that a workable data model for motion pictures has
been presented, the problem becomes how to fill it in with
real-world values: how is the data obtained? Two very
different approaches are explored: direct derivation BY the
computer, and human translation FOR the computer. The former
is a highly theoretical process, while the latter initially
seems like too much drudgery and bother to be worthwhile.

III.C.1. Direct derivation

III.C.1.a. Scene Recognition

Although a visual language of cinema is much discussed
by semioticists, the fact remains that deliberate non-verbal
communication of abstract ideas presents, at best, a
difficult problem for people, not to mention computer-vision
systems. Spoken language is much more clearly understood by
both categories of listeners, and written language better
still. If all descriptive data about a movie had to be
directly derived from a computer analysis of the video image
alone, computer- assisted video would be a unique case of the
blind leading the sighted. For how is it possible for a
computer to 'know' what exists on a piece of film in terms

that are meaningful to people? In reference to computer
vision:

> "It's much, much more complex than what people
> expected 10 or 15 years ago ... We don't even
> know how to represent a cat in the computer's
> memory. We can enter an iconic representation,
> but the computer would still need the power to
> generalize -- there are cats of different colors
> and sizes, tigers and house cats, even cartoon
> cats."
>
> > -- Daniel Sabbah, senior manager of
> > artificial intelligence systems,
> > IBM Yorktown
> > (NY Times, Sunday December 15, 1985)

Anyone who has ever closely watched a video waveform
monitor and vectorscope while a familiar scene transpired on
the picture monitor will surely be able to sympathize with
the machine trying to decipher a moving image. [see Figure
number 15] It is obvious that something is there, some
moving shape differentiated from a background. But making the
leap in conjectural analysis from "there is a mostly reddish
non-geometrical moving object in front of a static green
variegated background" to "a young woman with long black hair
and brown eyes wearing a red plaid shirt is walking through
her garden" is more the realm of science fiction than
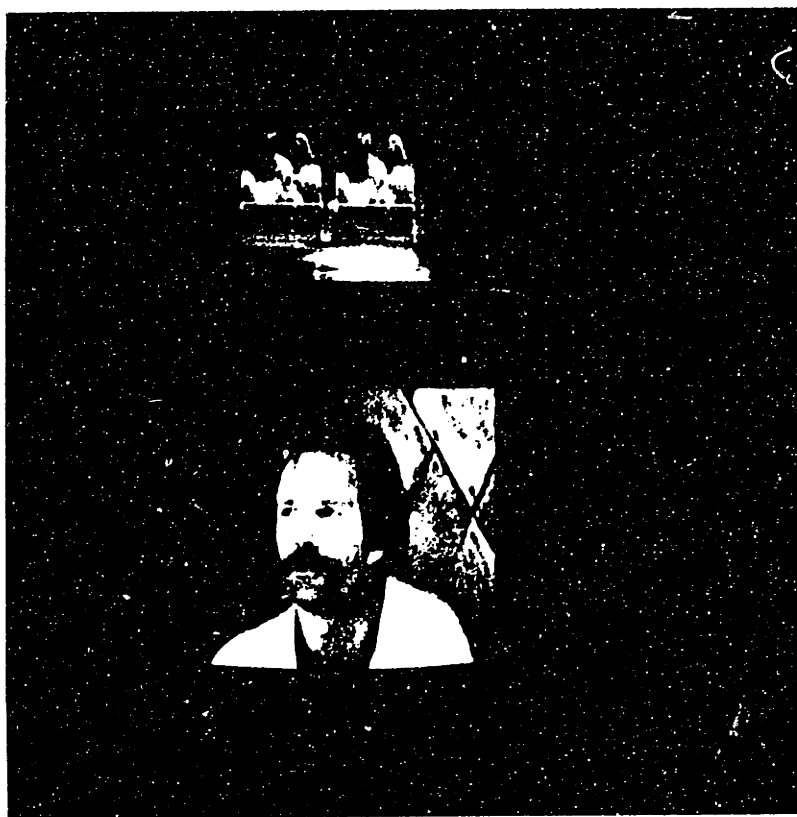
Figure 15. - A man and his waveform.

computer science, at least through the end of this century. Although robotic vision can be trained to recognize one machine part out of a hundred on an assembly line by linking a sophisticated spatial model to its interpretive process, such successes are due primarily to the limited contexts of the problem space. A robot trained to recognize bearing assemblies in a factory environment would be lost trying to pick characters from a movie image.

For movies have no fixed context; they can present any environment imaginable, and then some. Movie images are already abstractions; they contain two-dimensional data when the world is three dimensional, and they attempt to symbolize a reality we experience with five senses, not two. One has only to be reminded of the confusion or disinterest most household pets display when encountering a television image; they (apparently) do not perceive it as a 'window' into another world, but perhaps as a perplexing kind of light source. Most object-recognition systems rely upon a fixed lighting situation, from which shadows and edges give clues as to surface orientation, and a simple geometric model of the scene is matched against a database of 'known' objects [Havens]. These constraints are clearly difficult to meet in the context of typical movie images.

And so it remains our task to serve as mediators, feeding our hard-won interpretations of visual language to the computer on a textual silver spoon. But if such data

only has to be entered once by human hands - if it can then be permanently stored and conveniently accessed, and made a useful part of normal production procedures - perhaps we should not be so quick to turn up our noses.

In an attempt to derive direct visual information from movie materials by computer, a simple scene detector was created. This device continuously monitors a video channel, watching for the abrupt changes in picture content that usually signal a cut or edit. These events are noted in a master log, enabling the computer to separate video materials into semantic 'chunks' without user intervention or interpretation. For simplicity and speed, only luminance and chrominance are sampled at a small number of screen locations; statistical methods are used to infer the answer. The system employs a Grinnell 24-bit 'true-color' digitizing frame buffer attached to a Perkin-Elmer 32-bit super-minicomputer, and a computer-controlled one-inch videotape deck. Since the frame buffer requires three sequential digitizations (one each for red, green, and blue components), the source material is played back at half-speed to minimize color blur induced by motion. Nine evenly-distributed areas of one hundred pixels each are sampled for average color-component difference and average luminance difference. Each of the areas then 'votes' yes or no, depending on whether or not it has exceeded an established percent-difference factor with respect to the

previous frame sampled. If the number of votes is greater than the supplied threshold, the system concludes that a scene change has occurred. [Figure 16]

There are two input variables to the scene detector -- the 'percent difference threshold' and the 'number of votes threshold.' The noise level on this system appears to be around 8-10% difference found in at least 3 regions, below which almost every frame is flagged as a scene change. For hand-held documentary footage, a trial-and-error search for optimum values led to a 15% difference factor in at least 5 regions. Apparently, small voting thresholds do not remove local-motion residual effects; and smaller difference factors are too close to the noise threshold. Higher difference values presumably do not take into account the relative consistency of lighting ratios and color ranges inside a given scene, and so ignore many transitions. For the footage tested (a completed documentary film), detection of valid scene changes approached 90%. Unfortunately, false detections were also high, giving between 15% and 35% error rates depending on the scene content. The error rate is probably artificially high due to the unsteady nature of the camerawork; tripod- filmed materials should be less error-prone. Also, more sample regions and better statistical methods would undoubtedly improve performance.

Scene Detector Sampling Regions

```
1   2   3
4   5   6
7   8   9
```
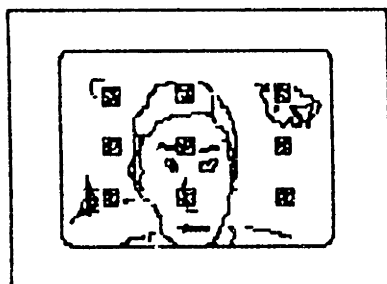
First Frame   **Second Frame**
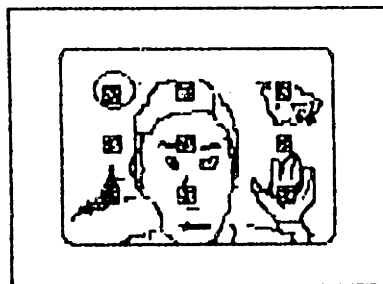
(frames shown with sampling grid superimposed)

Example: %difference threshold -- 17
                number of votes threshold -- 5

After analysis, only regions 1,7, and 9 exceed the
percent difference threshold. These regions vote
'YES'; all others vote 'NO.' Therefore the answer is
'No change detected yet.'

Figure 16.   -   The Scene Detector.

In discussing the above results with David Leitner of
DuArt Film Laboratories in New York, I found that he had
investigated the possibilities of using such equipment to
replace or augment the 'timer', the person who determines the
proper combination of filters in the film printer to
reproduce the best possible color in each scene. This is
known as scene-by-scene color correction, and is one of
DuArt's features for dailies and workprints. He commented
that although Dubner and other companies manufacture
reasonable scene detectors, the risk of making even one error
was more cost than the lab was willing to assume [Leitner].

The original impetus for creating the scene detector
was as an automatic input device for a computer-graphics
based video editing system. The idea was that the scene
nndetector would select frames from the video material to be
stored as representative samples of each shot. These
'picture cards' could then be shuffled to reflect the new
desired order of the shots, as discussed in Section IV below.
Originally, I thought it would be possible to find and store
only the beginning and ending frames of each shot, having
been inspired by a convention adopted by the Montage editing
system [Weynand].

After disheartening results, it seemed that perhaps the
scene detector could still be useful, by taking at least one
sample per shot (near the first frame) and then a number of

others if the content changed significantly. This method would take samples when the scene content was changing fastest. Thus a shot from a 'talking-head' interview might initiate only one sample, whereas a panning shot of a busy streetcorner would generate several. Finally, after reconsidering the nature of the editing process, it seems that the primary function of a representative frame is to jog the editor's memory by giving a strong visual connection to the material; hence the selection of such a frame should result from a personal choice rather than a statistical anomaly. This is a clear distinction between viewers and editors -- a sample frame will generally mean very little to a viewer until its associated segment has been seen, whereas the very same frame may be the strongest connotative link available to an editor. The sample frame is seen largely as an aid to recall, not a carrier of information in itself.

III.C.1.b. Textual Analysis

Linking a textual dialogue, or transcript, to its film counterpart represents another opportunity for the computer to derive uninterpreted, or (almost) direct, data from the medium to ascertain its content. Where might such data come from? In the case of a dramatic film, there will be very few differences between the finished script and the spoken dialogue; the data is already there. In the case of documentary materials, it is a common practice for the

editors to produce full transcripts of all the audio tracks as a first step. They then 'cut-and-paste' this document to create a "content cut" (formerly called a "paper cut" when transcripts were typed out) which contains all the information they wish to communicate. The remainder of the editing process becomes the search for a solution to various visual and aural problems encountered in the videotape version of the content cut. Once again, the data already exists. Even in cases where it doesn't, transcription is a relatively inexpensive process for the wealth of information it provides. In another 15 years, a computer may be able to directly transcribe audio materials; until then, it only costs about $50 per hour of material.

How might such data be used? A simple system might begin by creating a dictionary of every word mentioned, and linking the text segments to the video segments from which they came. A request to see the scenes where the word 'Apple' is mentioned might then bring up a news segment about the Apple Computer Corporation, or a cooking lesson on apple pies. These might be distinguished by requesting the scene in which 'apple' is mentioned within 5 words of either 'pie' or 'computer.' More advanced forms of lexical analysis could also be applied; the word class (noun, adjective, or verb) of each instance might be inferred from its context, and hence modifiers, objects, and subjects could be determined. Here

'Apple' is part of a noun phrase and 'apple' is an adjective modifying the noun 'pie.'

These techniques are well-understood though difficult to apply. Some inexpensive softwares, such as interactive fiction, rely to a great extent on the accuracy of such parsers. These programs expect to find short action phrases like "turn on the light" or "sit in the seat" [Aker]. Interpreting transcripts of normal speech is a much larger problem. In this case, humans have already performed the difficult (and hardware-intensive) task of connected-speech recognition; all that remains is the lexical analysis phase, which would not even have to be very accurate to derive useful data.

III.C.2. Human Interpretation

In developing the scene detector, I was struck by the amount of resources consumed in attempting to derive data that already existed. As a video editor, I know that almost every videotape produced has an associated edit list describing the exact location of every scene transition. Similarly, since documentary producers often create transcripts as part of the production process, why should anyone attempt to apply speech recognition technology to the finished audio? Why not analyze the transcript instead? Why can't such data be passed on, from maker to user? The

truth is that almost any data a viewer could desire has already been generated by the filmmakers; the problem becomes how to get the filmmakers to use the computer during these data-generation phases so that it may be easily published.

This thought began a quest to integrate the descriptive database procedures using dBASE III into the typical post-production sequence. The result is a fully-functional two-machine, keyboard-driven, frame-accurate, on-line editing system based on the IBM-PC, whicn works directly from dBASE III records. Here it is not only convenient to store additional data with each segment record, but easily worth the small extra effort for the time it saves in searching for and identifying one particular segment out of the hundreds involved in a typical production. EDIT uses two of the Ampex remote-control units mentioned earlier, a small single-key command vocabulary, and database browsing utilities using DBROWSE. The first version of this program is designed as a basic tool for conforming a master tape from a work-edit. The work-edit tape from the off-line session is entered into dBASE III format event-by-event, using burned-in time code numbers as the addressing reference. The program GENEDL is then run on this database to plug the correct master-tape addresses into reserved fields in each record. Creating the master tape is then a 2-key process: Load Next Event, Perform Edit.

Obviously the process would have to be extended to include the off-line editing to make it a truly integrated system. Specifically, the database should begin during the logging stage, when material is viewed for the first time and it is important to keep a record of what and where everything is. The contents of the descriptive database should probably include: segment in and out numbers; the major sequence or location of which it is a part; the main characters; the dialogue or an abstract description of it; thematic category; and shot description. These were found to be useful during the post-production on "Marital Fracture," which used the EDIT system exclusively. The final master database has been installed as the descriptive portion of a VideoFile installation system, and is working quite well except for a 'rippling' bug that placed a series of events out of their correct position by a second.

IV. Modes of Interaction, Presentation, and Access

Now that all of the methods are in place and a
descriptive data set has been delivered to the viewing
system, what functions will need to be included? These
problems are treated as issues of access, and are approached
on three levels: Physical Access, Access to Content, and
Structural Access.

IV.A. Physical Access Using Pictorial Conventions

There are three elemental pictorial objects to be
manipulated by the editing/accessing system, which represent
a relatively low level of interface to the material: Cards,
Strips, and Sheets. These fit nicely into the Projector
model described earlier, composed of segments and branches.
The intention is to provide efficient visual access to a
visual medium. At this, the Phyical Access level, the Maker
and Viewer are essentially the same in terms of their needs
for control of the video materials. These conventions would
therefore be useful in either a professional editing system
or in an interactive video installation system.

The sample frame is the foundation upon which the
pictorial conventions are based, serving as a reminder or
mental token of a larger movie experience. These picture-
icons are small, around 1/64th of screen size, in order that

a large number of them may be grouped on the screen at once. The representative frames would most likely be chosen by editors as part of the final post- production process, associating one picture-icon with each video segment in the final tape. In the case of videodisc, these icons could be published as digital data, ready for display as full-screen visual table of contents. For tapes, a special code in the timestamp might indicate that these are the sample frames for the segment within which they are encountered, allowing them to be digitized and stored on-site.

IV.A.1 Pictorial Conventions

IV.A.1.a. Cards

[see Figure 17]

Cards represent discrete logical units. A card may take the place of a segment (a raw piece of video), or of a sequence (a series of segment), or even higher abstractions (a decision tree, for example). The card is composed of a picture icon with a colored name strip pasted onto the top. The color of this tab might indicate the level of image hierarchy that the card represents. Blue could be for segments, green for sequences, and red for polylinear pathways.

[see Figure 18]
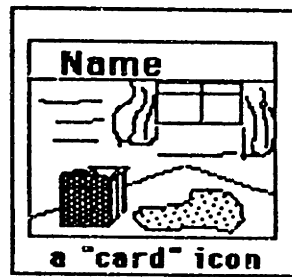
Figure 17. - A Card.
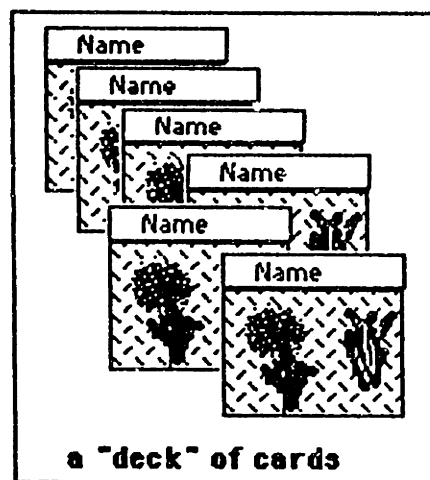
Figure 18. - A deck of Cards.
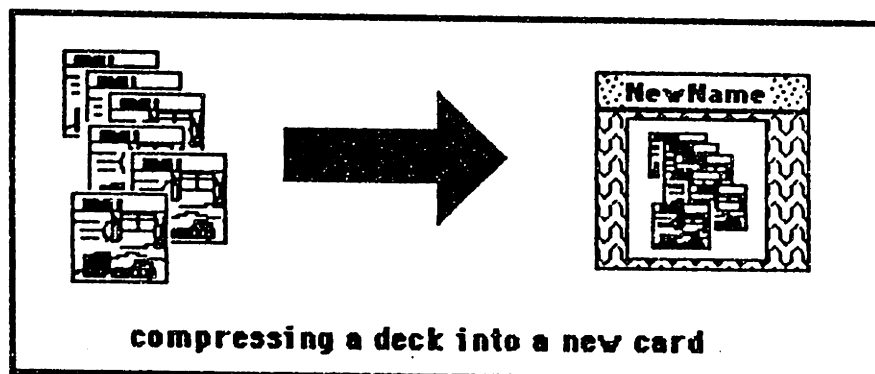
Figure 19. - Multi-level feature.

Cards may be aggregated by stacking them into 'decks,' which of course may be 'shuffled' in a pictorially analagous fashion. This seems to be a natural representation for a list, lending itself well to such processes as inserting, deleting, and moving elements. A program developed by Ron MacNeil of the MIT Visible Language Workshop, called "Groups of Lists," utilized just such an approach to manipulate abstract data objects. Cards may be picked from the deck by pointing to any visible area; they may then be moved within the current deck by inserting them between other cards or moving them to other decks. A deck may also be 'compressed' into a new card, which then represents the entire deck. This new card could be distinguished from its counterparts by the different color of its name strip, and also by choosing an appropriate picture icon.

[see Figure 19]

Naturally, this Meta-Card can be 'expanded' back to the deck it represents at any time. This zooming feature allows for quick transfers between levels of hierarchy, and gives simultaneous access to objects from different levels. For example, a sequence card could be inserted into a deck of segment cards. Since any kind of video unit can be made into a card, a generalized grouping and structuring mechanism has been established.

IV.A.1.b. Strips

[see Figure 20]

Strips represent continuous video footage, which includes both segments and sequences. This is the lowest level in the pictorial hierarchy, because it is closely tied to the linear-access physics of the medium. The strip is designed for clarity of visual interface, not for efficiency of representation. It may be 'scrolled' up or down. Strips are ideal for stepping through footage as one might do with a storyboard, or giving a concise view of a visual progression. So cards and decks of cards may be expanded to their strip version, to give users a sense of temporal context.

At some point (the ultimate editing or decision making point), no abstraction is possible or even desirable. The editor must have access to 'the real stuff' to make a pleasing juncture of image and sound, and the viewer must be able to see the actual video. The strip is intended to serve as a last recourse before that inevitable return to 'the real stuff.'

a sample "strip"
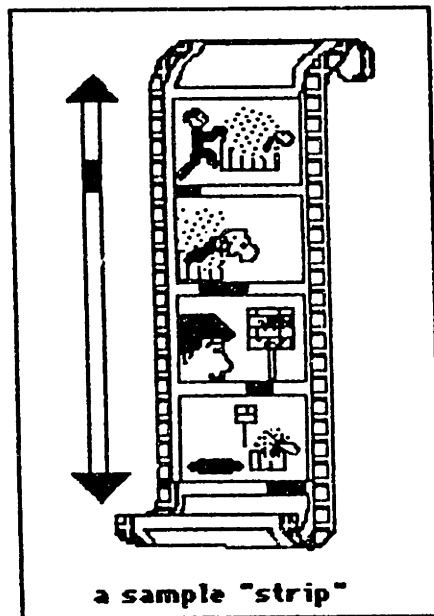
Figure 20. - A Strip.

"The AMT Icon"

from tape: #9

taken: April 18, 1985
IN   01:02:03:04
OUT 01:02:15:00
DUR 00:00:11:26

SYNOPSIS:

A shot of the AMT building from upper
Ames street, looking toward the river.
Notice the distinctive archway serving
as the gateway to the east campus.

Figure 21. - A sheet.

IV.A.1.c. Sheets

[see Figure 21]

Sheets contain textual data about cards or strips. They
may not be grouped or edited, only viewed. The sheet is
really a view of the object's record in the database, which
will contain whatever the editor has chosen as the most
important items to store. The goal is to provide the editor
with a facility for reviewing information which exists in the
same pictorial mode as the physical access methods of the
Card and the Strip.

IV.A.2. Graphic Implementation

The Pictorial Access conventions developed were
implemented on an experimental medium-resolution 640 x 480
pixels by 8-bit frame buffer called the YODA, developed by
IBM for use with a personal computer [Figure 22]. The YODA
contains a dedicated bit-slice microprocessor with microcode,
attaining high performance even on the relatively slow IBM
PC. One of its most intriguing characteristics is its large
'invisible buffer', a pool of image memory more that half as
large as the visible area, where pixel blocks may be
temporarily stored and then quickly copied back into the
visible region via a full complement of Bit-Blt functions
(routines that copy rectangular blocks of pixels). Perhaps
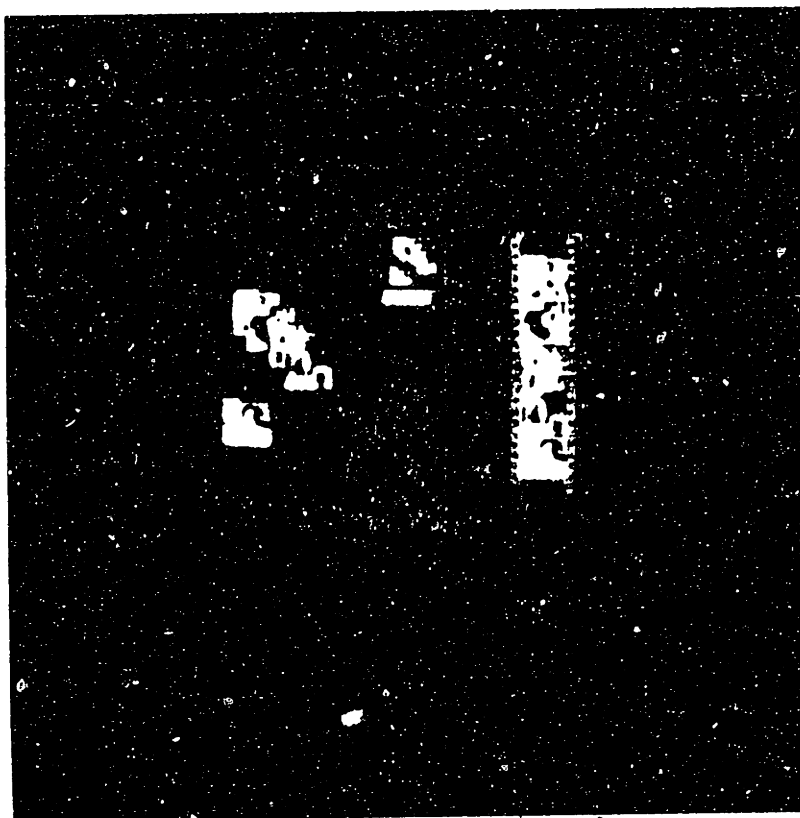
Figure 22. - Pictorial Access Conventions on the YODA frame buffer.
At left is a deck of cards being shuffled; in the
middle is a compressed version of the deck; at right
is a strip version of the deck.

its greatest asset is its ability to set 4-bit (16 grey-level) anti-aliased text fonts at an average .8 milliseconds per character [IBM].

Unfortunately, the YODA is incapable of digitizing from video, so the previously mentioned system involving a Grinnell 24-bit 512 x 512 frame buffer and a Perkin-Elmer minicomputer was used for the actual digitization. The Perkin-Elmer runs a version of the Multics operating system known as MagicSix, and code is written for a home-grown PL/1 compiler. GRABFRAME is given the timecode address of a desired sample frame, usually hand-picked for just such a purpose. The one-inch videotape is shuttled to the appropriate location and put into still-frame mode, and then a 3-pass color digitization of the video signal is performed. The NTSC signal from the deck is decoded to RGB, then low-pass filtered to remove color subcarrier residuals (due to the poor-quality decoder). Each of the color components is then amplified up to 1 Volt (peak-to-peak) by a line-driver Distribution Amplifier, and presented in turn to the NTSC video input of the Grinnell by a serial-control switchbox built by Keishi Kandori, MIT Research Affiliate to ABC, Japan.

If any motion-blur is detected in the digitized image, a field-filter program may be run to remove one of the offending fields (this takes about 7 seconds). The

full-frame image is then sub-sampled by SMALLPIX to 1/64th screen size (64 x 64 pixels) and saved in a special directory along with a 1-bit bitmap used for graphical directory searches. Note that smallpix does not currently average colors in order to save time; hence noisy images will most likely look even more so when subsampled. This entire process normally takes about 3 seconds from the time the videotape is put into still-frame mode.

Now the 24-bit subsampled image must be color-quantized down to 7 bits (128 colors). Although the YODA has 256 slots in its Video Lookup Table (VLT), a full 16 slots are used up by each text-color range; therefore, only 128 slots are used for images so that fully anti-aliased text in several colors may be displayed on the same screen. To allow any number of images to co-exist in the frame buffer at any one time, it was decided to quantize every image to the same 128-color map. This also saves a great deal of time in the color-mapping phase. The 128 colors were chosen as follows: 32 slots comprising an evenly-distributed gray-scale from black to white; 48 slots of one rotation through the IHS (Intensity/ Hue/Saturation) Color Cone at 33% saturation; and another 48 slots of another revolution at 66% saturation. After noticing some harsh artifacts of the low-saturation cutoff, it is believed that perhaps a better spread of 32 slots of rotation at 15% saturation, 48 slots at 40%, and 48 slots at 70%, may

lead to smoother color transitions. MAKECM stores these colors in a MagicSix file.

The program CRUSH is then run on the subsampled image to map its color-space into the 7-bit VLT produced by MAKECM. After much optimization, this process currently takes about one minute to perform. CRUSH simply finds the closest match in terms of RGB difference between each color in the original image and a color in the destination VLT. This data is saved in another MagicSix segment. Finally, the companion programs PUTFILE on the Perkin-Elmer and GETFILE on the PC are used to transfer the 4,096-byte image data directly to the PC in binary form at 9600 baud over a standard 8-bit serial interface. This data-file transfer takes about 15 seconds. Obviously, the path from original video image to a ready-to-load YODA data file is a less-than-direct route.

The subsampled images are used as the basic Cards, with colored name-strips pasted onto them with YODA Bi-Level fonts. Cards are dragged about the screen in real time by swapping image areas in and out of the invisible region, using a Hide/Display/Replace algorithm (Hide what you're about to cover, Display the picture there, then don't forget to Replace what you hid before leaving). Unfortunately, this leads to flicker problems which make images hard to recognize while they are in motion. The Strip is created by non-overlapping Cards set amongst bitmap images representing

a filmstrip. Five bitmaps make up the Strip's possible components: the standard Frame boundary, the UpScroll and the DownScroll (for the ends of complete strips), and the UpBreak and the DownBreak (for unfinished strips). All of these bitmaps were painted on the Grinnell frame buffer using a custom Fat-Bits routine which allows for zoomed-in lo-res painting and storage of registered bitmaps. The bitmaps were then shipped directly to the PC using PUTFILE/GETFILE.

The most interesting feature is the ability to simulate the 'shuffling' of a deck of Cards in real time. In this demonstration, a picked Card may be visually inserted into a deck between any two cards. Special masking techniques make the card appear to be behind one card and in front of the other [see Figure 23 for detail]. The overlapped region for each card 'above' the current card is added to a special card-sized bitmap. This bitmap is then used as a key-mask when copying the card into the frame buffer, preventing it from covering areas that are logically 'above' it.

As a sidelight, another use was found for the key-mask concept in creating instant full-color icons from video, for use as cursors or input devices or anything else. As an example, a frame showing a film reel was digitized, and a mask painted for it so that only the reel itself would show. This reel might then be used as a menu 'button' to initiate the playback function. Similarly, a stop-sign could be videotaped in the field and translated into an icon to inform

Inserting a Card Into a Deck

(visual editing)

Old List   **Card to Insert**

New Location
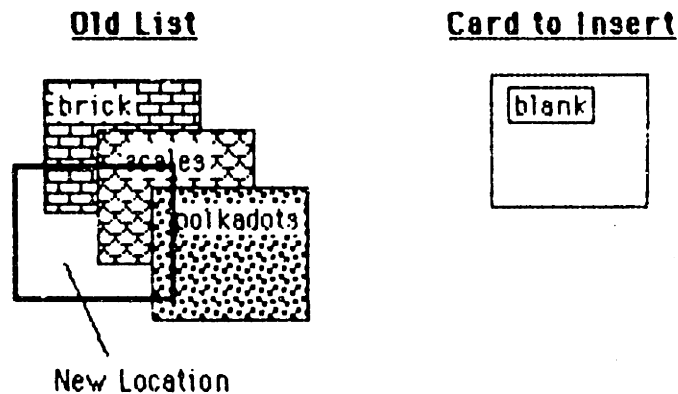
Insertion of card "blank" is between the card called
"brick" and that called "scales." A mask is made
composed of all the overlaps of cards ABOVE the new
level (in this case, "scales" and "polkadots") This
mask is used as a 'keying' image when placing "blank"
at the requested location.

Mask   **New List**

Result

Source
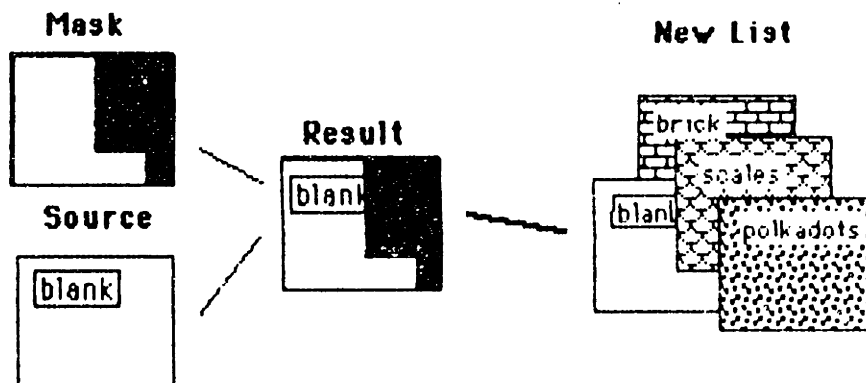
Figure 23. — Masking operations for decks of cards.

readers that a chapter was finished. One practical use
envisioned for the "Marital Fracture" installation is to
preserve masked icons which are close-ups of each character
in the video, so that viewers may quickly associate names
with faces even when they are not mentioned in the audio.

IV.B. Access to Content Using Textual Descriptions

The possibilities for deriving meaning from a critical
parsing of a video transcript were outlined earlier. The
transcript is one of the most important sources of data in an
installation system. It can clarify what may have been
mumbled or shouted, or may translate from one dialect or
language to another. It can be searched for mentions of
characters or events, or snatches of phrases half-rememberd.
Editors have found themselves saying " .. if only I could
find the shot where (X) talks about (Y) .. " ever since
'talkies' were invented.

The ability to provide this function was realized using
the FA and DC programming modules described earlier. ATOD
(for Add To Dictionary) adds a given text file (such as a
transcript) to a master dictionary, which keeps track of
every occurrence of every word in every file it has scanned.
FGETW gets the next word from the file, which is converted to
all capital letters and added to the dictionary. Then an
Instance structure is created, and attached at the tail of

the list of occurrences of this word. The file offset of the first character and the length of the word are stored in the instance structure, as well as a pointer to a structure of file information. Finally, next-word and previous-word pointers are set, and the instance count for this word is updated. This allows routines to look backward and forward a certain number of words in the file; a utility called DCSRCH searches the master dictionary for an occurrence of one word within a given span of another, such as the word 'apple' within 5 words of 'pie.' A list of files added and statistics about them is also maintained.

Since file offsets are stored with each instance, it is possible to recreate a span of text (from one word through another in a document) exactly as it appeared in the original source file, with spacing and punctuation intact. The span is fetched by reading in file data starting with the first letter of the first word of the span, and finishing with the last letter of the last word. One possible use was explored with SYNCTEXT, which displays text spans (also called text segments, or TextSegs) from a transcript file 'in sync' with the playback of the associated video segments. Although not fully functional, this module could become very powerful if the words were even loosely mapped to frame locations. A viewer might stop a video segment to inquire for more information; the computer could then calculate approximately which text span covered the frame at which input was given,

and infer the meaning of the user's action from the context of the span in which it occurred.

But working from transcripts can only get you so far; often what is important in a shot is what is NOT being discussed. For such subtle distinctions, it would be useful to have handy some interpretations of what is happening in each segment. Text which serves as a guide in this way will be termed an Abstract. Each record should have a comment field in which the editor explains what is 'important' in the segment, trying to use a small vocabulary of keywords. It should be much easier to search for imprecise content using such descriptions.

Another kind of Abstract data is the thematic category; each segment record might be given a list of theme-numbers from a master Thematic Index which the editors feel are relevant topics. Here a 'theme-number' is just a convenient and efficient shorthand for the text of the theme-name: theme number 17 might be "Home Improvement," while theme 42 could be "Solar Power." A video course on the economics of homebuilding might have a segment record with a field called THEMES, which contains "17;42". Such a scheme was demonstrated by Earl Yen in a program called THEMEVID, which asks for a list of themes the viewer is interested in, and then plays a short movie if enough matching segments are found.

IV.C. Structural Access Using Outlines

In contrast to the previous accessing modes discussed,
which dealt with the search for and manipulation of segments
and records, structural access lets viewers travel
information pathways which have been defined in previous
interactions with the system. A pathway could be a movie, or
a body of text, or a myriad of other things. Viewers are
able to modify these pathways to better suit their interests,
or they can create new ones and add them to a library. The
pathways together make up a tree structure containing all of
the possible interactions for the associated video. Also,
the tree can help to provide a sense of the 'level of
presentation;' popping 'up' levels presents the user with a
more abstract view of the information, while pushing 'down'
levels should generate more and more detailed information.

As a demonstration of the power of structural video
access, an Outline Processor called VTREE was developed,
adhering to the basic tenets of reconfigurable video as
heretofore stated. This program turns an ordinary text
outline into a video and text 'display tree', which may be
followed or appended to by any user. The concept of the
Outline Compiler was borrowed from Nardy Henigan's thesis
[Henigan], and the simple tree data-structure was inspired by
the tree Susan Wascher used in her thesis project [Wascher].

The intention is to provide a simple, generalizable method for structural access to the content of motion picture material - an active table of contents. Video segments may be accessed from videodisc or videotape (using the one-inch VTR and controller previously described).

A typical use might be for an instructor to provide an introductory VTREE outline to a broad topic represented in a case study, and request each student to create detailed outlines supporting a specific viewpoint. Suppose the given assignment is for a trial law class; the materials consist of video segments of the key witnesses and re-enactments of a crime, and a small library of supporting documents on-line. After a thorough exploration of the material guided by the instructor's VTREE outline, students act as the prosecution or the defense, trying to discredit the other side and establish an air-tight case by linking together video segments and text segments. These cases are then 'judged' by the instructor for clarity and thoroughness, attaching notes to the outlines where they are weak and need work.

This program takes a textual outline as input, which has a title, abstract and body. The body is a tree-structure of text lines with labels conforming to the standard hierarchy as follows:

I. Roman Numeral (Root level)

...

...

    A. Upper Case Letter (level One)

    ...

    ...

        1. Arabic Numeral (level Two)

        ...

        ...

            a. Lower Case Letter (level Three)

            ...

            ...

        ...

        ...

    ...

    ...

...

...

Once this structure is parsed and read into a VideoFile, the user can begin attaching and viewing segments from VTREE. Each node in the outline has a ShowList attached to it, consisting of an arbitrarily long linked list of text segments (TextSegs) and video segments (VideoSegs) to display. There is a simple mouse-driven interface which allows for quick tree-wise movement, bouncing up and down levels or returning to the root. The active display consists

of all the children of a node at a particular level; this keeps the number of displayed elements (and hence choices) down to a minimum. Thus if the Root Level displays I, II, and III, a 'push' down to the children of 'II' might leave the display as A, B, and C.

```
         The Tree    ------>   The Display


    I.
    II.                   I.

        A.                II.  ---PUSH--> A.

            1.            III.            B.

            2.                            C.

            3.

        B.

        C.

    III.
```

(an example of a node's ShowList)

Node II.A.1.: Crime and Punishment
SHOWLIST: TextSeg 3, TextSeg 47, VideoSeg 18, VideoSeg 5;

Node names and their levels are displayed with the currently selected node highlighted, until the user pushes a button on the mouse. At this point the function buttons appear at the bottom of the screen for selection of the

action to apply to the current node. Functions currently
implemented include:

Show:

Displays the ShowList of the selected node

All Of:

Displays the ShowList of the selected node
and those of all its descendants (in
depth-first tree-wise order)

Up:

The new level is the parent level of the
current level

Down:

The new level is the first-child level of
the current level

Root:

The new level is the root level

Append:

Attaches a video or text segment to the tail
of the selected node's ShowList

Quit:

Leaves the program

Display functions consist of the ability to show just what
exists at a given node, or to show a node's contents and the

contents of all of its descendants. This tends to encourage the use of the higher-level labels as more 'abstract' presentations of what lies underneath them, in a kind of abstraction-ladder treatment. Also, it seems that the first item in each ShowList might want to be a brief TextSeg which describes its intention; this way viewers have something besides the label names on which to base a decision for displaying or exploring.

The attachment function works by first inquiring if it is to attach a VideoSeg or TextSeg. If it is text, the program opens the requested file in full-screen mode, and gives roaming capabilities to the mouse (UP a screen or DOWN a screen). The user marks a text segment by pressing the left mouse button over the text IN-point, which is then highlighted. The first push of the right button after an IN-point is marked defines the OUT-point as the character over which the mouse is positioned; the screen is then cleared, the TextSeg attached, and the user returns to the previous location in the VTREE [see Figure 24].

VideoSegs are attached using an interface to dBASE III records instead of directly to the disc player. This was done to encourage the use of editorial descriptions embedded in the records, which will in many cases greatly speed the search for content not previously viewed. In the test case implementation using "Marital Fracture", the same database

Picking a Text Segment

Text Segment In-Point

Now is the time for all good men to come to the aid of interactive video. The lazy round videodisc jumped over the vertical interval. A switch in time saves nine cuts. Ask not for whom the video glitches, it does so for thee.

| UP | | DOWN | | QUIT |

Text Segment Out-Point
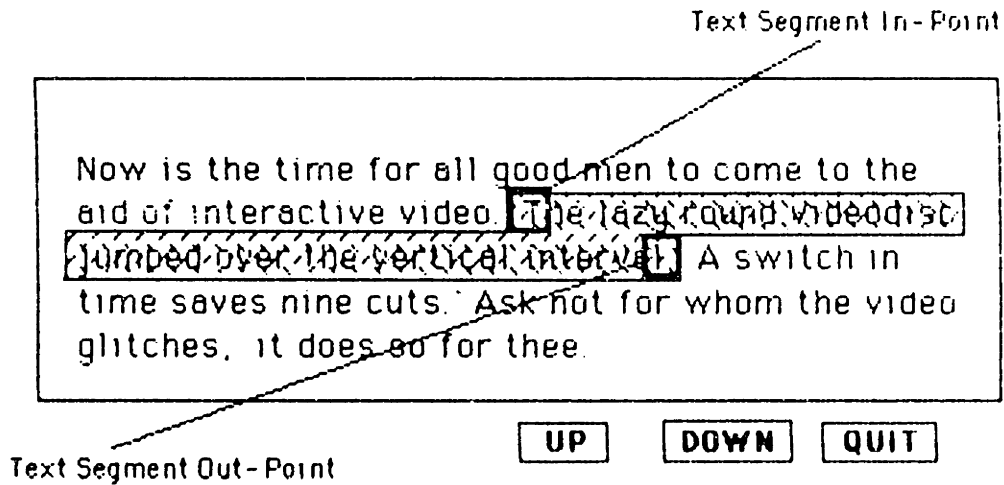
Figure 24. - Definition of a Text Segment.

Picking an Outline Segment

(using future "Where?" function)

```
I.  First Level
      A  This is A          ——————— Current Level is "I A"
      B.  And this B
      C  And so on
II  Second Level  ————————— New 'click' changes
                             level to "II"
```
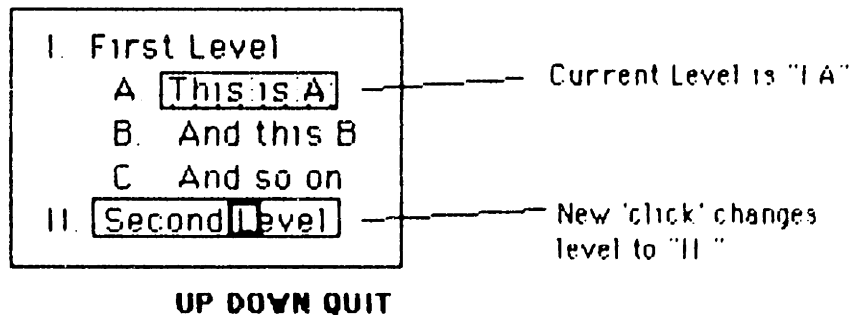
UP DOWN QUIT

Figure 25. - Changing levels in VTREE.

may be used for tape or disc, since this is the database which was used to create the master tape from which the disc was pressed. When the VideoFile is initialized, it finds the first VideoSeg and queries the user as to which database record fields it should interpret as the video in and out-points, and whether tape or disc is the display medium. In the future this information will be stored with each VideoSeg. When a VideoSeg attachment is requested, the last previously-accessed record is displayed on the screen, and the user is given browsing capabilities using the cursor keys on the keyboard (Up a record, Down a record, GoTo a specific record, and Select a record). After selection, the record may be viewed prior to attachment. Then the user is returned to the previous location in the outline.

The power of this simple concept is in the simplicity of its operation - there are only seven functions that accomplish everything - and in its applicability to almost any subject matter on tape or disc. It certainly fulfills the stated goal of providing a table of contents for video resource materials for new viewers, and a means of previewing segments, and is a novice-oriented system. And there is a clear and comfortable distinction between the three levels of movie data discussed: addressing (in and out points in the records, which apply to disc or tape), descriptive (the database records), and associative (TextSegs and the outline tree in general). Most happily, it is both easy and fun to

modify, something that few interactive video programs can claim. Users should get the impression that the building of outlines is a collaborative venture, and that they should improve and add to them if they can.

One of the limitations is the need to travel up and down tree levels without knowing what is there before you go. What is required is a mode of direct, or world-view, access in addition to the currently provided tree-wise access. This will soon be accomplished by the addition of an eighth function button called "Where?". The 'Where?' function will allow direct access to all levels of the outline as well as giving a much-needed sense of context when perusing the tree structure. This will be done by letting the user roam the text file from which the outline was originally parsed. The text labels' in and out marks are already stored in the VideoFile, and so it is very easy to discover which label the user is 'inside' when the mouse is clicked [Fig. 25]. When the 'Where?' function is selected, the outline file will be opened as a full-screen window, with the currently active node highlighted. If the user clicks inside a different label, that node becomes the current node and the user is returned to tree-walking. Otherwise the current node remains unchanged. This way a quick 'click' can give an instant sense of place in the overall outline.

Another easily-remedied fault is the inability to edit the ShowList once it is in place; new segments are always

attached at the tail of the list. Obviously there needs to be a set of list-editing functions available in the 'Attach' module. Potentially the Pictorial Access Conventions I mentioned earlier could be used to perform this task. Also, it would be very useful to add the PATTM module discussed earlier to the database browse, so that records may be more directly accessed by their content. Similarly, a set of logical connectives would be helpful in searching - "match occurances of A or B, but not C." But perhaps the most interesting modification of all would be to allow different sorts of objects to be attached to the nodes, instead of just ShowLists. Suppose graphics could be attached, or functions, or rules. What if other Outlines could be accessed from an outline tree? This could truly be the foundation of a video library.

V. Conclusion

A method has been described whereby descriptions of video materials may be composed by their makers and delivered to their users, and the benefits of such a practice explored. Viewers would be provided with a consistent and uniform interface to movie materials regardless of medium, format, or hardware system, and would regain a measure of the explorative independence associated with other media such as books. This interface would most likely take the form of an outline-based mode of structural access, with computer graphics simulation for physical access. The marriage of database techniques to motion picture content is found to be an effective method for viewer-initiated presentations.

Television has long been accused of being a passive medium, unlike the active pursuit of reading a book. Books provide a personal experience: delighting and informing at the pace, time, and place the reader desires. Specific pieces of information are indexed through standardized resources such as the Table of Contents, and libraries are maintained by cross-referencing schemes such as the Reader's Guide to Periodicals and the Dewey Decimal System. However, video and film provide a richness and exactitude of experience that words cannot approach. Unfortunately, these media have been presented in Projector-fashion, and the accompanying ineffeciencies of personal access to the information they

provide are well-known. Through the use of computers and developing technologies, the best of both worlds can be obtained.

The softwares developed in the context of this project remain largely experiments, simulations of what might be achieved in the best of possible worlds. The techniques of database management in the world of video production and distribution, however, are applicable now, and represent an advance in the way information can be linked to video materials. Perhaps the adoption of such an approach would be the first step towards the delivery of Reconfigurable Video as it has been explained here. An integrated delivery system which combines these techniques to present a personalized accessing and cross-referencing method for video would demonstrate the validity of the concept, and inspire further development.

References

[Aker]
        Sharon Zardetto Aker.
        "Wake Up to Adventure."
        in 'MacWorld,' May 1985.

[Anderson]
        Gary H. Anderson.
        "Video Editing and Post-Production:
        A Professional Guide."
        Knowledge Industry Publications, White Plains,
        NY, 1984.

[Ashton-Tate]
        dBASE III Reference Manual.
        Technical Memorandum.
        Ashton-Tate Inc., Culver City, CA, 1984.

[Atre]
        S. Atre.
        "Database: Structured Techniques for Design,
        Performance, and Management."
        John Wiley and Sons, Inc., New York, NY, 1980.

[Brown]
        Eric Brown.
        "Optical Publishing."
        Master's Thesis, MIT, February, 1983.

[Ciciora]
        Walter Ciciora and Gary Sgrigroli and William Thomas.
        "An Introduction to Teletext and Viewdata with
        Comments on Applicability."
        in 'IEEE Transactions on Consumer Electronics',
        July 1979.

[Danna]
        Sammy R. Danna.
        "Stereo Audio For TV - What Now?"
        in 'EITV,' February 1985.

[EECO]
        "The Time Code Book."
        Technical Memorandum.
        EECO Inc., Santa Ana, CA., 1984.
        3rd Edition.

[Forester]
Tom Forester, editor.
"The Information Technology Revolution."
MIT Press, Cambridge MA, 1985.

[Gano]
Steve Gano.
"Forms for Electronic Books."
Master's Thesis, MIT, June 1983.

[Gerstein]
Rosalyn Gerstein.
"Interpreting the Female Voice: An Application
of Art and Media Technology."
PhD Thesis, MIT, February 1986.

Also:
"Marital Fracture: A Moral Tale."
Interactive videodisc project, produced at the
MIT Film/Video Section.
Excerpted on the "Elastic Movies" disc, 1984.

[Havens]
William Havens and Alan Mackwirth.
"Representing Knowledge of the Visual World."
in 'Compute,' October 1983.

[Henigan]
Nardy Henigan.
"Software Tools for the Graphic Designer."
Master's Thesis, MIT, June 1983.

[IBM]
YODA 0.2 Programmer's Guide.
Version 3.00.
IBM Corporation, Yorktown Heights, NY, 1985.

[Kawaguchi]
Shigetoshi Kawaguchi and Nobuhisa Nagata and
Hideo Ikegami.
"Automatic Editing-Control Data-Collecting System."
in 'SMPTE Journal', May 1983.

[Lattice]
Lattice dBC Library Functional Description Manual.
Technical Memorandum.
Lattice, Inc., Glen Ellyn, IL, 1984.

[Leitner]
David Leitner.
DuArt Film Labs, Inc.
(conversations).

[Neuman]
W. Russell Neuman.
"Television and American Culture."
MIT Report, Communications Technology and Research.

[Newsweek]
"The Age of Video."
Newsweek, Dec. 30, 1985.

[Norton]
Peter Norton.
"Inside the IBM PC."
Robert J. Brady Co., Bowie MD, 1983.

[Owen]
Bruce M. Owen.
"Television Economics."
Lexington Books, Lexington MA, 1974.

[SMPTE]
Society of Motion Picture and Television Engineers.
Memoranda published over PROJECT CONFER, an
electronic conferencing service.

[SONY]
SONY LDP-1000 Interface Manual.
Technical Memorandum.
Sony Corporation, 1983.

[Strong]
William S. Strong.
"The Copyright Book - A Practical Guide."
MIT Press, Cambridge MA, 1984.
Second Edition.

[Wascher]
Susan Wascher.
"CONFORM: A Context for Electronic Layout."
Master's Thesis, MIT, June 1983.

[Weynand]
Diana Weynand.
"An Editing Alternative."
in 'Videography,' April 1984.