If you could see what I hear: Editing assistance through cinematic parsing

by

Natalio Carlos Pincever

B.S., Electrical Engineering
University of Florida
Gainesville, Florida
1989

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

Signature of the Author [illegible]

Natalio C. Pincever
Media Arts and Sciences Section
May 10, 1991

Certified by [illegible]

Glorianna Davenport
Assistant Professor of Media Technology
Thesis Supervisor

Accepted By [illegible]

Stephen A. Benton
Chairperson
Departmental Committee on Graduate Students

If You Could See What I Hear: Editing assistance through cinematic parsing

by

Natalio C. Pincever

Submitted to the Media Arts and Sciences Section, School of Architecture and Planning, on May 10, 1991 in partial fulfillment of the requirements of the degree of Master of Science at the Massachusetts Institute of Technology

Abstract

The proliferation of camcorders has made it possible for everyone to be a film director. However, the editing process is still reserved for a few specialists who have access to sophisticated equipment. In order to offer a low-cost tool which encourages novices to participate in the process, we need to better understand how the power of computers and digital signal processing technology can help to detect basic cinematic modules. The described system combines approaches from different disciplines in order to probe the audio track to find said modules.The system assists the user in the editing process by suggesting potential editing points.

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

Acknowledgements

I would like to thank my family: el viejo, la vieja, Becky, Adrián and Mayra. I couldn't have made it this far without them.

I'd also like to thank the following people: John Watlington and Barry Vercoe for reading the original proposal, and help in shaping up the ideas; Carlos Reátegui and Derrick Yim for their super-human effort; Dan Ellis, for all the tools; Andy Hong, for his all-around niceness; Michael Hawley, for all his useful suggestions and ideas; Chris Schmandt, for giving me a chance; Thomas Aguierre Smith, mi buen amigo, for all the great arguments and collaborations; and the rest of the people at the Media Lab, for making this place what it is: an effervescent avalanche of ideas.

Finally, I'd like to thank Glorianna Davenport, my advisor, my teacher, my friend, for just about everything.

Table of Contents

1. Introduction

A Computational Approach to Designing the Home Moviemaker's Assistant

Most, if not all, home movies remain unedited. Why? For one reason, electronic editing equipment has until recently been too expensive for consumer acquisition, and it is only recently that consumer VCRs have been built with editing capabilities and computer ports. Another reason that home movies are not edited is that editing -telling a concise, coherent story- is hard work and time consuming. In order to overcome the typical reluctance of users to edit, we must identify why that reluctance exists, and design a system that is easy to use, fun, and encouraging, since editing is hard and time-consuming. More people would edit their footage if aided by a computational system which possessed enough knowledge about the editing process to be of assistance.

The system described herein is designed to assist the user with the editing process, based on some knowledge we can program. This particular system looks at the audio track as a content indicator. The audio track is used to parse the content of rushes, which are the first generation stream of information. There's enough information encoded in the audio signal to obtain a great number, if not all, of the shot changes. This is achieved through the analysis of background noise levels, found to be a very reliable cue in determining shot boundaries. Several disciplines are merged in this design, all contributing different elements and useful concepts. Based on observations of raw footage, several heuristics were developed to map changes in the audio signal to what happens on-screen. These heuristics are then applied to the raw audio data, in the form of a soundfile, which is the digitized audio track of the video to be analyzed. Further analysis is then performed on this audio data, using well-known techniques. The results are then presented to the user, who interactively logs the shots found by the system, with the changes she deems necessary.

1.1 The concept of an Assistant Editor

Even though most "How to do it" manuals encourage users to edit their footage, the majority of home movies remain unedited. As editing equipment becomes available, there will be a necessity to bridge the gap between the user, who possesses no knowledge on the editing process, and the equipment. This gap can be bridged through the "enhancement" of the required tools, with the help of a computer. This computer would then become the novice editor's aide, or "Assistant Editor."

The concept of the "Assistant Editor" is simple. The purpose is to create a system that will assist in the editing process, that is at once easy to use, fun, and encouraging. Such a system must possess knowledge of the editing process, as well as the capability to use this knowledge in order to assist the user. What are the ideal characteristics/attributes of such a system?

- It should be non-intrusive, meaning that the user should retain the controlling role in the process.
- It should be interruptible. The user must be able to, at any time, override the system's findings and decisions.
- It must be easy and fun to use, in order to overcome the user's reluctance to edit the material.
- It must be as interactive as possible.
- It must offer valuable assistance.

These are the design guidelines which will allow the system to gracefully interact with the user. As this interaction occurs, the user will learn more about the editing process, and more collaborative interactions can evolve.

2. Background

Ever since the early days of cinema, there have been attempts to create a portable camera that would transport the moviemaking experience into the home. The first attempt was the Biokam in 1899, which was a 17.5 mm system. It was later followed by Pathé's 28 mm system in 1912, and several others. In 1923, Kodak introduced the 16mm system, which was perhaps the first true home movie system. However, the first real revolution in home moviemaking came with the advent of *Super 8* , which became extremely popular. Users were true filmmakers in that they worked in all aspects of the moviemaking experience: shooting, developing, editing (splicing the film), and exhibition. The low cost made this medium accessible to a wide audience, who kept editing benches in their garages and basements. "Super 8 has turned filmmaking into a medium which is not unlike writing. Everybody is doing it. Filmmaking is fast becoming a viable means of personal communication," declares the enthusiastic Lenny Lipton [Lipton 75].

The hand-held video camera, or camcorder[1] , is far more ubiquitous than Super 8. The VHS camcorder was introduced in the US 11 years after the black & white reel-to-reel consumer Portapak. These compact, shoulder

[1]Commercial video equipment has been available for some time. The first video camera was created in 1925 by John Logie Baird; the first videotape recorder was sold by Ampex in 1956.

mounted units, imported from Japan, arrived in this country in 1981 (Panasonic and RCA first, with Sony close behind with a different format). The first video 8mm units came in 1984 courtesy of Kodak, and this format is gaining popularity over the bigger VHS units, although the latter are still quite popular.

The proliferation of camcorders has made it possible for everyone to be a film director. Unofficial statistics show that there's a camcorder in approximately 10 percent of American households, which amounts to more than 9 million camcorders. These figures are on the rise, aided by the increasing ease of use of newer models. Despite this proliferation, home video editing is still unusual. It is quite obvious that most "home movie directors" do not have formal schooling in the cinema arts. Such lack of formal training would lead one to assume that each user has a unique method, perspective, theme, and score. However, the few studies which have been made observe certain trends. While we can use these studies to isolate indicators, readers should realize that the studies we have found make observations based on broad demographics.

2.1 Home movies

Home moviemaking is perhaps the most overlooked form of visual communication. Considered a minor form, the only notable types of documentation on the subject are "How to do it" manuals. Home moviemaking is unique in that it is a form of private and personal communication, yet it is associated with mass communication technology.

A limited anthropological study shows that the communication aspect of home movies has not been controlled by technology; rather, it is controlled and structured by social prescriptions and limitations. The study also shows that most users completely disregard advice given by "How to do it" manuals[2]. Contrary to what the manuals say, most users never seem to plan their shots, never think about camera techniques, and are extremely reluctant to edit the footage. There also appears to be a certain uniformity in most home movies. The majority of home movies depict people, usually close friends or relatives. This is also true in the exhibition phase, since most movies are shown to close friends and relatives, usually people that were in the movie itself. Another striking observation is that the off-camera events (i.e. the actual shooting) are handled by the head of the household, generally the older male. It should be pointed out that this study only took white, middle-class families as samples. Cross-cultural studies, such as one involving Navajo Indians, show some interesting results in that the subject matter, as well as the filming techniques, change drastically [WA 70].

[2]Chalsen describes the situation in detail, and includes several accounts and interviews with a population sample[Chalsen 75].

This study brings us back to the question: why do home movies remain unedited? We underscore our earlier response: a) Cost of equipment;
b) difficulty of the thought task. If we want to encourage editing, we need to assist the user in every way possible.

2.2 What is in a soundtrack?

Computationally, it is less expensive to parse audio than it is to parse video, since video requires a more complicated, multidimensional representation. There is strong evidence to suggest that separate audio and video servers will be used in the digital world [Applebaum 90] [Arons 89], as each has a unique set of attributes, such as granularity of representations [DPAS 91]. It is then useful to identify potential applications for such servers. If there is enough information in the audio track to extract shot boundaries, then why bother trying to obtain this information from a more complicated, multidimensional representation?

A shot is comprised of one or more frames which are *generated and recorded contiguously* and which *represent a continuous action in time and space*. A shot is defined by two numbers, uniquely indicating the location and duration of the shot: frame-in and frame-out. Clearly, the original shot generated in the camera may be redefined many times during the editing process. Frequently a passage of sound is recorded along with the shot (either on the same or on a separate medium). We refer to this sound as *synchronous sound*. Synchronous sound is made up of meaningful dialog and ambient sound. If this synchronous sound track is not physically bound to picture (as in film or digital modes where separate picture and sound servers will be used), it must carry some identifying tag which will guarantee that it will maintain its synchronous relationship with the picture despite adjustments which may occur during editing.

A different category of sound also present in some movies is *wild sound*. As theaters became outfitted for projection, presenters thought to accompany movies with live music, most frequently piano; music was selected to enhance particular reactions of the audience. These live accompaniments were free form, that is they were not specifically rehearsed or synchronized to the action. This is an instance of wild sound [DPAS 91].

As soon as it became technically possible to lock sound to the moving image for projection, sound became the ubiquitous partner of the motion picture. While the audience perceived this track to be synchronous with the picture, most release tracks for narrative film were created in dubbing studios and foley pits during the post-production process. (Even today most sound tracks for feature films are created during post-production.)

In traditional cinema, the goal of the editor is to create a seamless experience. Shots are assembled to form sequences which play on the viewer's ability to build cognitive structures. Sound is often used to blur distinctions at the edges of shots, sequences and scenes. As a final step in cinematic post-production, after the editor has fine-tuned or "locked picture," a sound editor is frequently employed to "clean up the track," embellish the feeling of being there by inserting sound effects, and to design audio transitions. In addition to synch sound, a track providing explanatory commentary or music may be added to make information more explicit, to heighten the perception of an action, or to set a mood. This process of adding special effects and music links back to the expressive methodology of early filmmakers.

As described above, a shot is the basic building block of cinema. Typically, synchronous sound accompanies original shots in home movies. The editor's job is to create sequences by joining shots. Any editing system which encourages logging or other segment manipulation begins with the required definition of shot units. To assist the user in the editing process, an editor's aide *must be able to determine shot boundaries.* We believe that these shot boundaries can be identified by parsing the synchronous audio track. The user can then proceed to build sequences, based on the building blocks (shots) the system provides.

Audio tracks of home movies

Home movies audio tracks are essentially composed of the same elements already described for all movies: ambient sound and meaningful dialog. The latter sometimes includes some narration, usually performed by the same person doing the shooting. This differs from commercial movies, where the narration is usually generated later, in the post-production stage. In commercial movies the dialog is sometimes altered in a process known as Automatic Dialog Replacement (ADR), also referred to as looping.

Yet it is in the ambient sound where we find the most interesting distinction. In commercial movies, this element plays a secondary role. Most ambient noise is undesirable, and is eventually eliminated. In fact, many reshoots are caused by unwanted changes in ambient noise (like a plane flying overhead). Such noises are extremely difficult to edit. The only type of ambient noise generally consciously respected is what's called "room tone," which is used to create the mood of the space in which the action is supposed to occur.

Two kinds of noise can be recognized: narrow band noise, and broadband noise. Narrow band noise has limited spectral content; thus, it is not difficult to edit out. All that is needed is a filter designed to clean up the particular bands in which such noise is present. Broadband noise, on the other hand, has components throughout the frequency range. It would be impossible to build a filter to remove this noise, since it would have to remove all frequencies, leaving no sound at all!

Ambient sound in home movies has all these elements present, from cars going by to hand adjustments on the camera. No home moviemaker is going to reshoot her son blowing out the birthday candles because someone in the back of the room was sneezing. Room tone becomes more than just ambiance: it becomes the specific attribute of the time and space in which the shooting is being done.

3. Previous Research

Looking for a solution to the problem of assisting home movie editors, we can examine several traditional disciplines: Film Theory, Artificial Intelligence (AI), and Digital Signal Processing (DSP). There are several useful concepts from all these disciplines which can be incorporated into the overall system design: concepts from film theory reveal the shot as a primitive; DSP offers useful transformations of audio data into more useful forms, which will be easier to analyze; some research in AI offers a methodology for understanding and utilizing the results of the analysis.

3.1 Video Editing systems

The intent of an editor's assistant requires the intervention of computation at three or more levels: Segmentation at the lowest level, Representation, and Story Structure at the highest level. The most basic form of editing is splicing film. A very old process, it still has a romanticism all its own. "It's a great tactile experience to handle the very film you shot with your own flesh and blood fingers" [Lipton 75]. "Film is a wonderful physical medium. You can take a piece of film in your hands and stretch it out to get a sense of it's duration; you can hold it up to the light to see what the images look like. You can also get it accidentally caught on the edge of your editing table, break it, scratch it, and get it stuck to your clothes. Film is the noble savage of audio-visual media" [Rubin 89]. No other medium comes close to offering the flexibility of film, with all it's problems. Computer editing systems must be able to recapture that flexibility, and, hopefully, that charm as well. There have been several systems built which attempt to fulfill the above-stated goals: Sasnett's system at the segmentation level, Bloch's at the representation and structure levels, and Rubin's at the structure level.

An interesting system which attempts to segment video material was built by Sasnett [Sassnet 86] in 1985; the system coded changes in image content, such as luminance and chrominance, for nine points spread through the screen. An algorithm was devised in which the different regions of the screen, represented by the points, voted on the possibility of a scene change. These votes were the result of inferences made based on colorcomponent differences and average luminance differences. Clearly, these would never show zooms, pans, or shaky movement, for none of these would cause changes in either chrominance or luminance. The votes would then be analyzed, and a scale was used to check the results. Vote patterns would then be matched to this scale, which was then used to decide whether a shot change had occurred. Sasnett called his system the "Scene Detector." The results it yielded were encouraging, capturing 90% of valid scene changes. However, false detections were also high (somewhere between 15 and 35%, depending on the scene content), particularly fast pans and zooms.

Another system, inspired by Giles Bloch's work (see below), was implemented by Rubin, the *Constraint-Based Editor* [Rubin 89]. This system creates sequences automatically, using rules derived from film theory concepts on continuity. These rules create transitions based on five classes of temporal relationships and three classes of spatial relationships. The material is broken down into shots manually, including as much information as possible. This information is kept in a log, which forms the basis for a database, in which each field represents a particular shot attribute: description, characters, angle of view, etc. This database is then manipulated by a set of programs that, functionally, are very similar to those operators implemented by Bloch. The most interesting one of these is the one Rubin calls the *doctor* , which behaves like a "grammar-checker" that uses simple information on some of the database fields to spot and cure potentially poor transitions.

Artificial Intelligence: Rule- and Knowledge-based systems

Rule-based systems are those that use a set of predefined rules already programmed into the system to make decisions. The key issue in building such systems is the formulation and articulation of the appropriate rules.

Knowledge-based systems, as the name reveals, have pertinent knowledge already stored in such a way that the computer can make use of it to generalize or to perform some set of functions. This knowledge can adopt two forms: rules, which make knowledge-based systems a superset of rule-based systems; or common knowledge. The hardest problem to tackle when building knowledge bases is selecting an appropriate representation: how is this information to be represented so that a computer can understand it. A common representation of such knowledge is that of a heuristic, which is defined as "task-dependent information"[Nilsson 80]. Such heuristics can be thought of as "guesses" used to make inferences as to what is actually happening. See "Methodology" for a more detailed explanation.

These concepts can be used in the design of the system. Based on observation of raw footage, heuristics that define what a shot is can be formulated. These heuristics are then programmed into a rule-based module that will examine data from the audio track, and apply these rules where necessary.

One of the most interesting computer editing systems using these concepts was built by Bloch [Bloch 88] . This system builds sequences based on what can be called a "cinematographic language," as well as techniques borrowed from Natural Language Understanding. Shots and sequences are represented with a number of formal and semantic slots: actions, setting, actors, look, position, and motion. Based on film theory concepts, sequences are built by making inferences based on relationships among these slots from different shots. These inferences, however, can produce some changes in the meaning of the shot based on its context. The implementation consists of five operators, each working at a different level:*Choice* , *Construction* , *Appreciation* , *Correction* , and *Projection* . The Choice operator determines the type of segment needed; the Construction operator recognizes classic forms of editing, through the use of specialists for some of the frames described earlier; Appreciation and Correction also use specialists for other frames to check the joining, with Correction making changes as needed; and Projection deals with the actual showing of the sequences.

Agents and agencies

Minsky defines an agent as "Any part or process of the mind that by itself is simple enough to understand-even though the interactions among groups of such agents may produce phenomena that are much harder to understand"[Minsky 85]. This concept of an autonomous unit with domain-restricted knowledge (it knows its function and purpose, but nothing else) can be easily transported to other domains, and implemented on a computer. In the proposed system, it is possible to define agents that will perform certain low-level duties, which together will form what are called agencies. Agencies are defined as "any assembly of parts considered in terms of what it can accomplish as a unit, without regard to what each part does by itself"[Minsky 85]. Such a breakdown allows for a robust implementation, as well as modularity, for each agent has only a particular task to perform. For example, one agent can observe and identify certain parameters in the audio track. Another higher-level agent can take that information and analyze said parameters. Taking these agents together, an agency called "Change finder" can be built, whose function would be to find changes in a certain parameter existing in the audio track.

Although these ideas affected the original conception of the system, they were not used in the current implementation specifically. Rather, this theory suggests how the basic research of this thesis might evolve. Once such agents can be built, it may be possible to extract even more useful information from the audio track.

3.2 Audio Processing

As it has been established earlier, there is enough information encoded in the audio signal to obtain a great number, if not all, of the shot changes. Several audio techniques can be used to obtain the information encoded in the audio track. There are several useful elements of signal processing which may be incorporated into the system design. Of utmost interest are the analysis tools that the field has to offer, for they transform audio data into a more useful form. Statistical properties of the audio signal, such as the average power of a block of samples, may be computed for analysis purposes. These will be useful in the first stage of parsing: time-domain analysis of the signal. They can tell us where changes in the power of the signal occur, which generally indicates drastic changes in background noise levels. This is analogous to "reading the VU meter" on an audio mixing board: as the overall sound level changes, so does the VU meter indicator. With the help of heuristics, such processing can give an indication of the type of changes happening: a rapid increase followed by a rapid decrease is usually a thump, caused by someone shutting a door, or dropping something heavy on the floor, or grandma falling off her chair.

Another powerful tool is the Fast Fourier Transform, or FFT. The FFT allows spectral (frequency-domain) analysis of a signal. Such analysis is useful for filtering operations, recognition and detection of signals whose spectral content has been predefined. FFTs are slow and computer-intensive, yet they are very accurate, as frequency domain analysis provides a large amount of information. The FFT may be replaced by simpler transforms. There are several techniques available to analyze the spectrum of the signal once it has been obtained. Of interest are average spectrum and average amplitude (sum of mean-squared spectrum). These techniques allow for the analysis of changes in the spectral content of a signal, as well as changes in the power distribution. The latter is usually a good indicator of changes in background noise levels. It can be used in the following manner: a template is created with the spectral characteristics of a sample window, which is then compared with the following window. If there are considerable changes in the power distribution, then it can be safely assumed that the spectral content of this second window is different, thus finding changes in the spectral content of the signal, rather than finding changes in signal power. A combination of both average amplitude and average spectrum should suffice to capture enough changes in signal content to find the shot boundaries.

4. Project Description

4.1 Methodology

Through the observation of raw footage, it is possible to come up with a set of generalizations which, after successful testing, could be robust enough to be programmed as heuristics into a system. Such observations show:

- Difference in background noise usually means a different shot.
- Loud, singular noises should be reviewed by the editor, since they usually represent unwanted sounds, like a tap on the microphone.

These two assertions look deceptively simple. Yet, if robust, they offer a sufficient framework to build the assistant editor on.

Localizing Shot Boundaries

Preliminary research shows background noise level to be a powerful cue in recognizing shot boundaries. Thus, the first step is to be able to recognize such differences. If the difference in background noise is greater than a pre-established threshold, then it is safe to assume that this is a new shot.

Some digital audio processing will be required in order to prepare the audio data for the system to use in making intelligent decisions. Both time- and frequency-domain analysis will be performed on the digitized audio track.

Time-domain analysis will provide information used to determine differences in background levels, as well as for identifying peaks (usually caused by unwanted loud noises). Frequency domain analysis will be used to determine whether loud sounds are actually unwanted noise (non-harmonic waveforms). It may also be used for more sophisticated processing (like voice detection).

As stated earlier, the shot is the fundamental building block for storytelling using visual media. As mentioned earlier, It is fairly obvious that the editing tasks will be greatly simplified if such building blocks are identified for the novice editor.

Unwanted Noise Reduction

I have already established that loud, singular noises should usually be removed. Making decisions based on this rule is slightly more difficult, since it cannot be determined as reliably as the difference in background noise. Not all loud noises are necessarily unwanted: a mug being set on a table, a door slamming, someone falling off a chair. Thus, user input will be necessary to resolve this dichotomy. Assuming that an unwanted noise has been found, how do we remove it? One way to do this would be to use a template for what background noise level should be for the current shot. The unwanted noise is cut out of the track. The space created by this removal is then filled with background noise, looping if the amount of space is larger than the stored template. The soundfile then has to be re-recorded onto the audio track of the source material, using timecode for synchronization purposes.

The problem which now arises is how to choose this background noise template. Ideally, the computer should be able to do this. However, this requires the formulation of a set of heuristics. The cognitive mechanisms used to separate sounds are still unknown, and there is intensive research in signal separation. If such an algorithm is found, then it will easily integrate into this system, both for unwanted noise reduction and for content analysis.

Content Analysis

Using audio as an indication of patterns of content could prove quite useful. The audio track can provide information that otherwise would be extremely difficult to parse. Take, for example, a funny incident: the family is gathered around the table, grandma falls off her chair, and everyone laughs. In the audio track, this appears as a constant signal level, followed by a sharp increase, a sharp decrease, and a rise of a fluctuating signal level. The constant level is caused by the mumble of the family; the spike is caused by grandma falling; and the fluctuation is caused by everyone laughing.

As stated earlier, there's no way of telling whether this spike was just a thump, or was it something more interesting. Given our cognitive understanding of human behavior, the presence of laughter is an indication of a funny incident . If it is possible to know what laughter looks like, then we can parse it, and thus tell that something funny happened. Therefore, it is not desirable to remove such a spike from the audio track.

Another interesting possibility is to develop a library of templates, or "spectral signatures" of different sounds, to keep for later comparison. It might be possible to find content elements through template matching. For example, assume that the system has stored the spectral content of a fire engine roaring by. By checking through the soundtrack for occurrences of this spectral pattern, the system can tell that within a certain segment, there's a fire engine. This in turn suggests that there is a fire. Eventually, parsing could lead to the complete automation of the logging process, provided there was enough memory for storing patterns and enough computing power to perform the comparisons.

A third method for extracting content information would be to run a pitchtracker through the audio track. This would yield pitch information, useful in finding occurrences of human speech. The human voice has several frequency components, which are multiples of the fundamental frequency F_0. This fundamental frequency determines what pitch we hear the voice in. There is a certain range of frequency values that F_0 can take. For adult males, this range is from 80 to 240 Hertz; for adult females, 140-500 Hz; for children, anywhere up to 600 Hz [Handel 1989]. It should be pointed out, however, that these values are in no way absolute, making this a difficult feature to implement. However, identifying changes of voice in dialog sequences could be useful. Another difficulty is that several musical instruments have frequency components within this range, making it practically impossible to determine whether a particular sound was a voice or an instrument without additional information.

Mechanisms for User Input

We are working toward an assistant process; humans may be required to do some error correcting. The interface to the system is an important issue. A graphical interface will be provided, in which the user will make editing decisions using a mouse, clicking on buttons. The design guidelines used for this interface are quite simple. The goal is not to design a flawless interface full of features. Rather, the interface is kept to a minimum. The exploration of the rules and heuristics that make up the parser are far more important to this work than having a nice interface.

Most of the interaction is through dialog boxes, which appear on the screen during interactions. These dialogs have buttons, which the user clicks on to make decisions. The only other element of the interface is the shot logger, in which the user, once the appropriate shot boundaries are chosen, types in a description of each shot.

The user must be provided with a method for overseeing the changes being made, as well as participating in the decision-making process. For example: a loud, single sound is heard. It is difficult for the program to know if this was an accidental tap on the microphone, or if it was the example stated above: grandma falling off her chair. Clearly, we do not want the system to cut the sound out if the latter was the case. Thus, the user must be prompted for a decision, accompanied by playback of the sound in question, as well as the original video segment.

4.2 Implementation

4.2.1 Hardware configuration

The hardware configuration of the proposed systems calls for a computer, a video Hi8 player with SMPTE timecode, a video monitor, an audio amplifier, and speakers (or headphones). The following figure shows a diagram of the hardware implementation.

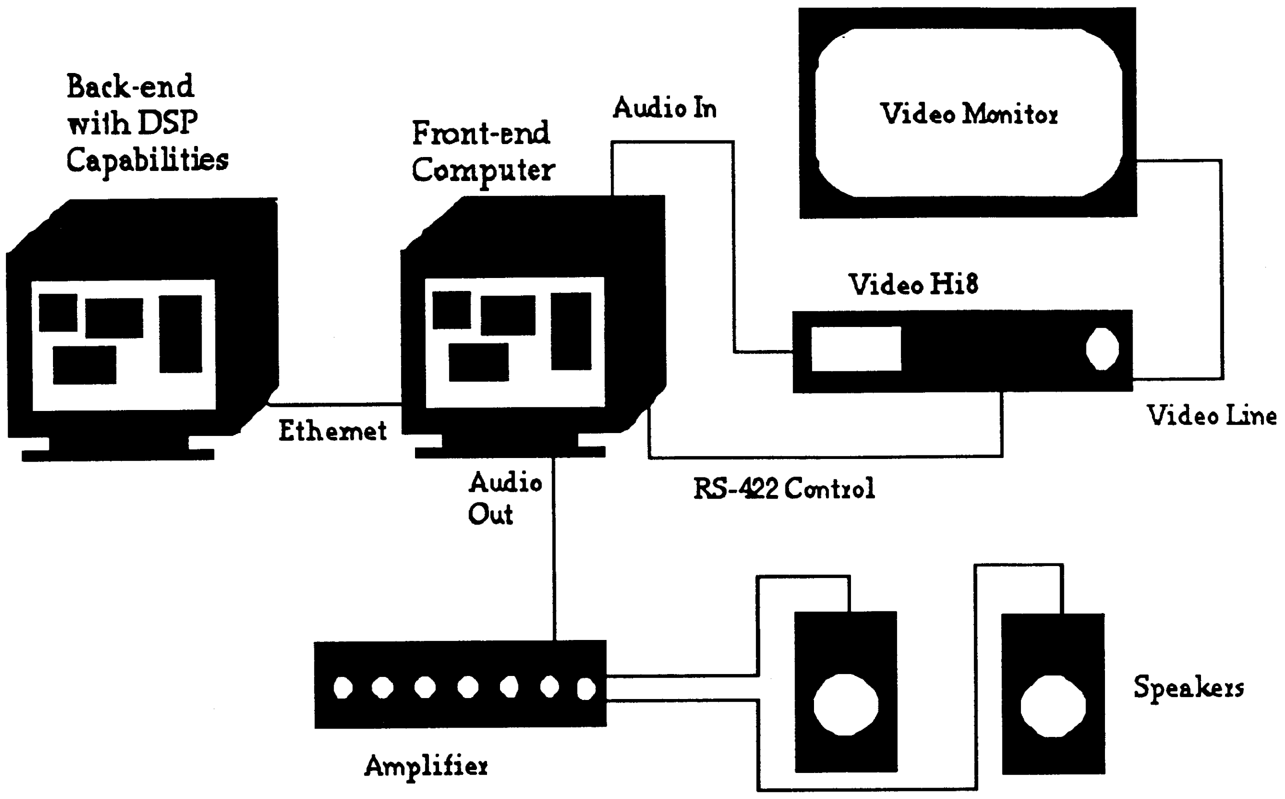


Figure 1: Hardware configuration

The video signal from the Hi8 player will go straight to the monitor, to allow the user to keep track of what is in the video material. It is in the audio track where all the processing is done. The audio from the video player is digitized, to allow the digital signal processor to analyze the signal content.

The original design called for a NeXT computer, due to its signal processing capabilities. As that machine was unavailable, a compromise between speed and ease of use had to be made. The solution achieved was to link two different computers via Ethernet. The front-end machine is a Macintosh IIci with the DSP added as a NuBus card. The card used is the

Digidesign "Sound Accelerator", which comes with an audio editing environment (described in the software section). The most computer-intensive signal processing, calculating the FFTs, was done on another computer, a DECstation 5000, which was chosen due to its speed and availability. Because of disk-size restrictions on the latter, the first level of parsing was performed on the Macintosh computer, passing only the chunks of information requiring further processing to the DECstation.

The video player used is a Sony EVO 9800. It is a Hi8 player (format used by most camcorders, thus it is the format in which home movies are recorded) with several necessary options, such as SMPTE timecode and RS-422 external control. These features allow accurate search and playback functions, necessary for successfully selecting edit points (frame-in and frame-out for shots).

4.2.2 Software

The following is a diagram of the different software modules of the system, an appropriate mix of off-the-shelf video playback and audio digitizing equipment and new software. Descriptions of each module follow.

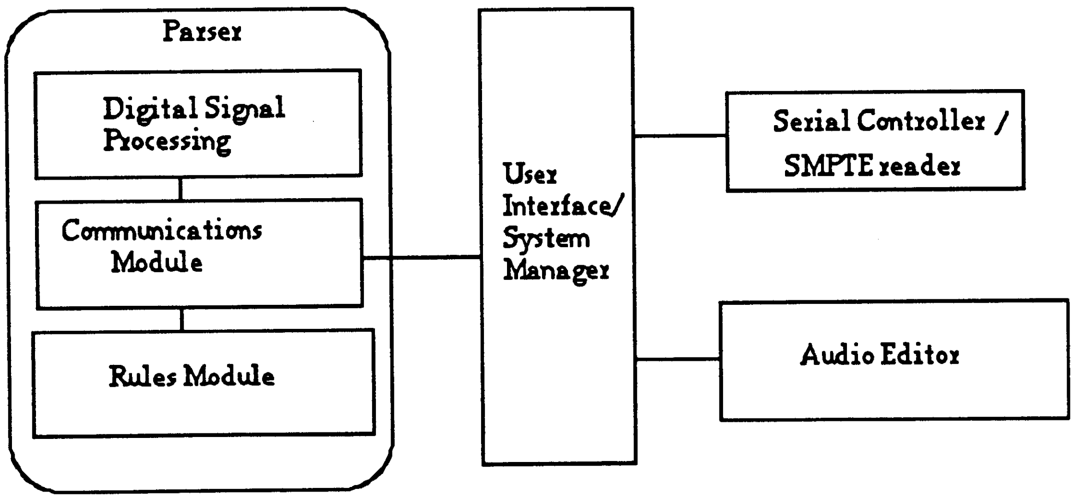

Parser
Digital Signal Processing
Communications Module
Rules Module
User Interface/ System Manager
Serial Controller / SMPTE reader
Audio Editor

Figure 2: Software implementation

<u>Serial controller</u>: The function of this module is to control the video Hi8 player via a serial interface. It will execute the basic control commands: Play, Stop, Rewind, Fast Forward, Record. The SMPTE timecode reader will do just that, read the timecode from the video player. It will also store the numbers in a buffer for quick access. This module will also work together with the serial control module, in passing SMPTE numbers to the latter for frame-searching purposes.

Audio editor: The audio editor used was the software which accompanies the Digidesign "Sound Accelerator" NuBus card, "Sound Designer". This software allows for complete recording and playback capabilities, as well as the ability to display an amplitude representation of the soundfile (see figure below). See the "Sample Interaction" section for usage information.

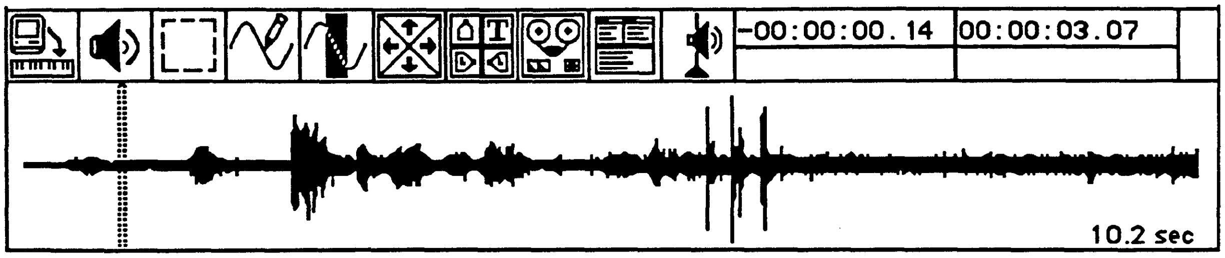


Figure 3: Soundfile representation

Careful thought was given to the possibility of using stereo soundfiles; these were discarded mainly due to the fact that most home movies are recorded with a single microphone, thus recording the same information on both left and right channels, duplicating the computing power necessary to analyze them. Also, recording them in stereo duplicates the size of the soundfiles, which creates serious storage problems; a stereo, 16-bit soundfile sampled at 44.1 KHz will occupy about 10 megabytes of storage space per minute of sound.

Parser: The parser examines the audio stream looking for certain kinds of changes; it has two different modules: the digital signal processor, and the rules module. The digital signal processor does the actual parsing required. To achieve this, it performs several tasks, such as Fast Fourier Transforms, Linear Predictive Coding , and statistical analysis. The calculation of the Fourier Transforms was done remotely on a different machine (a DECstation 5000), through a TCP/IP socket connection. The soundfiles were modified slightly before transmittal: the byte ordering was swapped, and the data packaging modified[3].

The rules module takes the form of heuristics, which make certain "guesses" as to how to interpret the information extracted by the signal processing module.

The parsing itself is done in different layers, each giving different information that could not be extracted from the others. The rules domain interprets the information obtained by the digital signal processing module,

[3]The format used is a hybrid between the IRCAM soundfile format and that used by NeXT. See [Ellis 91] for more details.

and apply the rules programmed into it, or heuristics, in order to make intelligent decisions on the editing process.

The first layer consists of time-domain analysis of the sampled waveform. The soundfile is divided into chunks, each being the equivalent of a frame of video. The number of samples representing a frames (1470) is obtained by dividing the sampling rate (44.1 KHz) by the number of frames in a second (30 for NTSC video). The root-mean-squared amplitude (RMS) is then obtained for each block. Next, the parser assumes that the first RMS value is the initial background noise level. The parser looks for changes in the average RMS value that are larger than the tolerance level. This tolerance number is a constant in the code. After several test runs, it was set to.26 or 26%. The parser keeps track of two numbers: the current minimum and the current average. The current minimum is set every time that an RMS value for a video frame goes below the previous minimum and is within the tolerance. If not, the parser assumes that there is a possible shot boundary where the signal level dropped, and sends a segment of three frames before the "boundary" and three frames after, where it assumes that the signal levels out to this new setting.

The current average is dealt with in a similar way, except that it is not set to the new value; instead, it is averaged with the previous value. Essentially, this creates a window of two average RMS values, and all the numbers in between are ignored. If they are below, they are treated as explained above, and if above the window they are averaged with the previous current average to update it (keep track of rising background noise). That is, if they are within the tolerance level. If not, the parser counts to see how long it takes before the RMS values return to the current window. If it takes longer than the frame tolerance (currently set at 43 frames: several values were tested, and this one gave the best results), it assumes that a shot boundary occured somewhere in between. Thus, it sends three frames before the initial value and three frames after to the second parser. At this point, the signal level may drop back to an allowed level, in which case the system continues. The sound may drop below the minimum, in which case it assumes a boundary in between the initial value and this final value, and then passes it to the next stage. Every time that the system encounters a possible shot boundary, it resets both the current minimum and the current maximum to be the next lowest number.

Using heuristics based on observation of raw footage, some preliminary conclusions are then drawn from the observation of these mean values. Here is where the first decision as to what is a shot change is made. For example, let's assume that the mean increases sharply, to then decrease as sharply as it increased after a short period of time (a "spike"). The heuristic in this case would be that this was a thump, caused by a slamming door, an accidental tap on the camera microphone, or a loud knock. Another example would be a smooth increase followed by a smooth decrease. The heuristic for this case could yield several results: a passing car, an airplane flying overhead. None of these would qualify as a shot change, since they are transitory changes in background noise levels. There are other cases which could possibly be shot transitions, but the information available at this stage of processing would be insufficient. For example, a smooth increase in average amplitude. This could mean that a car drove into frame and stopped, that someone turned on a noisy device (like a fan or a computer), or it could also represent a smooth pan (which could mean a shot transition). In these cases, a second layer of processing is required. This second layer consists of frequency-domain analysis, after performing a Short Time Fourier Transform on the soundfile. Analysis done at this level consist of standard signal processing techniques such as average spectrum, which is the equivalent of the mean analysis performed on the first layer, and average amplitude (sum of mean-squared spectrum). These two analysis schemes detect changes in the energy distribution on all frequency bands, thus capturing changes in the nature of the signal, not changes in the amplitude. This second-level processing is only performed for cases that are ambiguous at the first level, meaning that a clear distinction cannot be made simply with amplitude changes.

Interface/System Manager: This module allows the user to control the whole process through point-and-click events. It also lets the user to monitor what the different modules are doing, as well as ask for user input when necessary. This interface allows interruptability at every stage of the process, giving the user complete control, and thus allowing the user to relinquish as much control as desired.

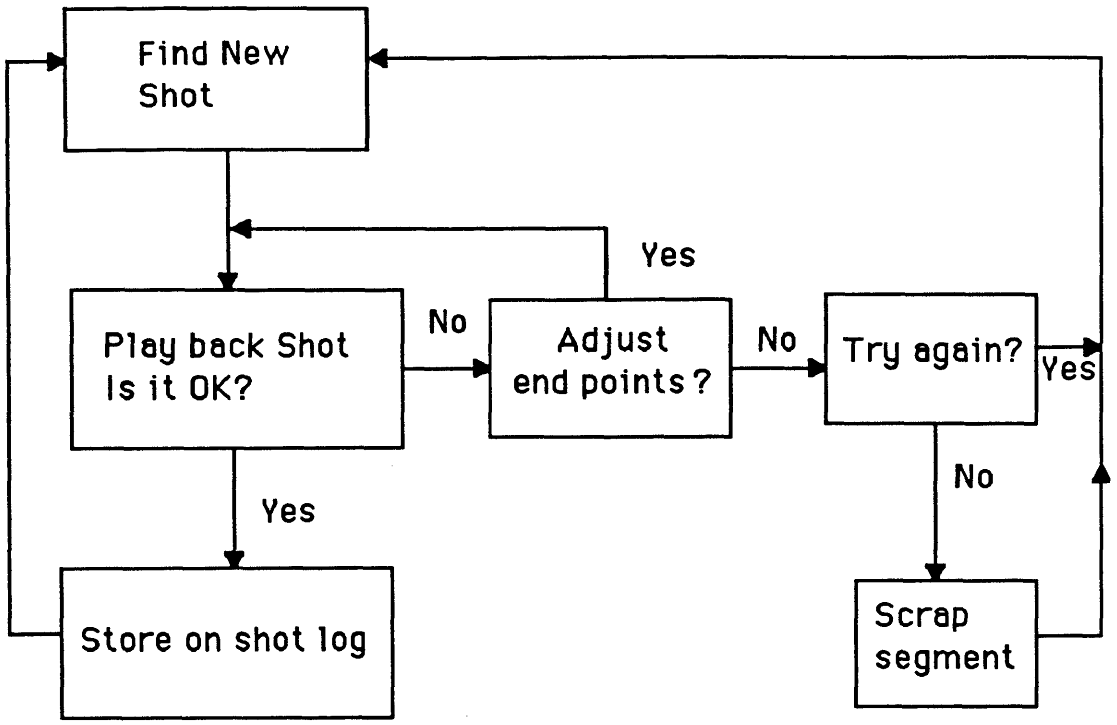


Figure 4: Interface interaction

As can be clearly seen in the diagram, the interface permits the user to scrap a segment if she thinks it is of no use. This is a quite useful feature, since it allows the user to eventually go back to either change the material, or to log it manually. If the system is not performing the way the user wants it to, this feature will let her override it altogether. The other interesting feature is the logging ability: the user is able to log the video as the shot boundaries are being found. This is also extremely useful, since a full log of the material will be available immediately, and the shot descriptions should be very accurate, since the user is will describe what he is actually seeing.

The user-interface module is also the higher-level controller. It's function is to schedule tasks to perform, pass information to and between modules, and obtain user input through the graphical user interface.

4.3 System performance evaluation

The system's performance was evaluated by running several different types of shot transitions. In this section, I'll describe the system's performance in evaluating five different transitions, which are a representative sample. These were chosen tp reflect a range of problems and difficulty, the first being the simplest case. The transitions were as follows:

- Simple transition, from black to a shot (nothing -> something).

- A shot change in which there's a human voice present up to the transition point.

- A pan in a noisy environment.

- A shot change in which there's music in the background in both shots.

- A shot change that occurs right after a car drove into scene.

These transitions, except for the first one, challenge the parser in that the background noise levels are in flux at the moment of the transition. Simple, straightforward transitions that the parser has no trouble dealing with are of no illustrative value. The following sections will deal with performance for each sample transition.

<u>From black to noise</u>

This is the easiest transition to pick up. The following figure shows the evaluation using the "dspbench" tool [Ellis 91]. The top third shows an amplitude representation of the soundfile, the middle third shows the spectrum for the particular thumbnail position, and the bottom third shows the phase.

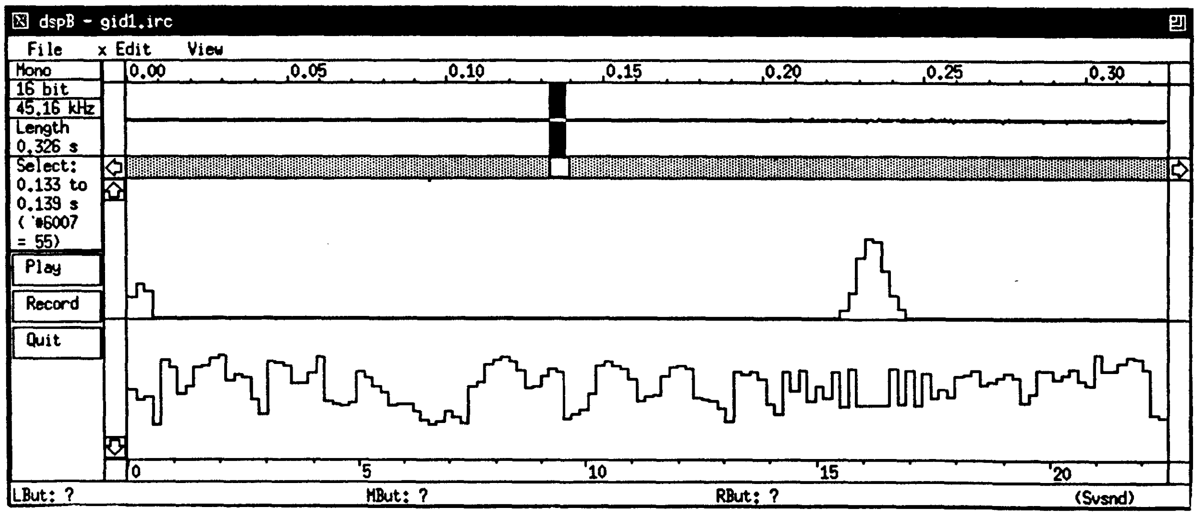


Figure 5: From black to noise-Before the change

As it can be seen in the figure, even though there's supposed to be no sound, there's some high-frequency components present (at the right of the figure). I have been unable to trace this noise: it may either come from the camcorder heads, or from the digitizing equipment.

The first parser accurately picks the transition point, since the change in amplitude is fairly obvious at about 0.20 seconds. The second parser then receives a block of samples consisting of three frames before and three frames after the possible transition point. The FFT is performed, and the results of figures 5 and 6 are produced. Figure 5 shows the spectrum before the transition, and Figure 6 shows the spectrum after the transition.

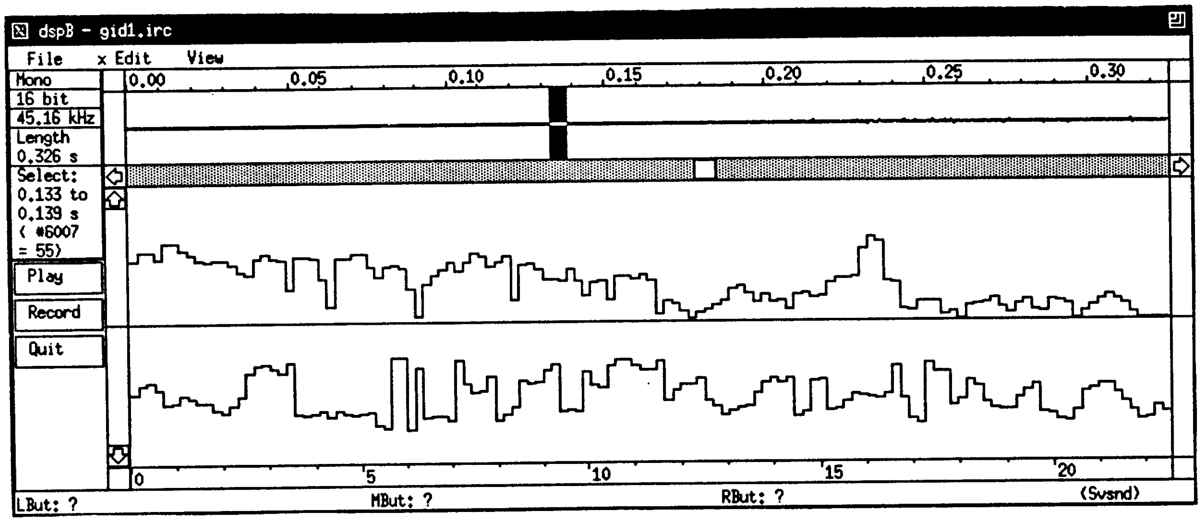


Figure 6: From black to noise-After the change

It can be clearly seen that the spectrum changes drastically from one figure to the other. The second parser evaluates an average frame, and then calculates differences. The differences go up drastically after the 0.20 second mark, correctly yielding the shot transition at that frame.

Voice up to the transition

This one works fine, even though there was a voice leading up to the transition, thus confusing the parser by showing constant changes in frequency response. The first parser again accurately picks the transition point, since the RMS amplitude of the signal goes up higher than the 26% tolerance level. Again, three frames before and three frames after are sent to the second level parser. The spectrum is obtained, and analyzed. Figure 7 shows the spectrum before the transition.

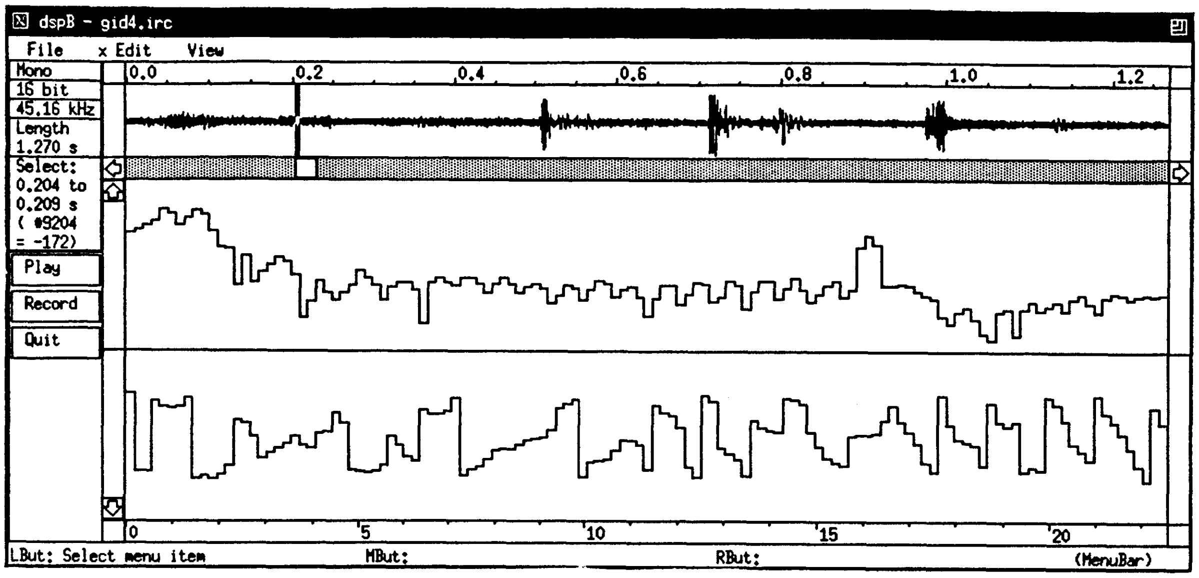


Figure 7: Voice up to the transition- Before

As stated in the previous example, the parser proceeds to calculate differences from the sample frame. The differences increase with the spectrum shown in Figure 8.

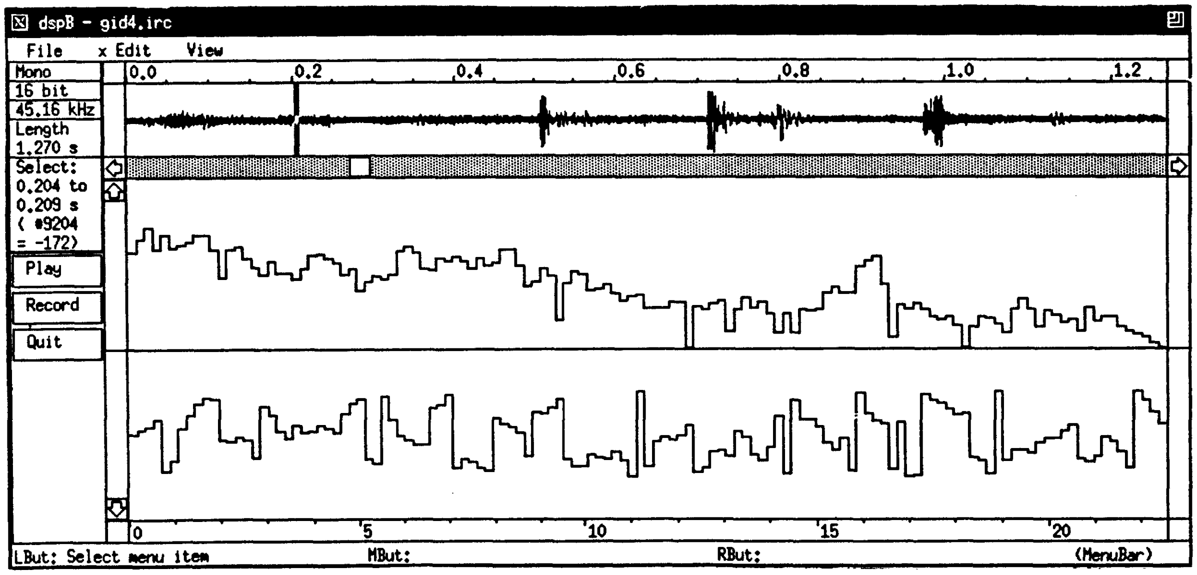


Figure 8: Voice up to the transition- After

Again, it is fairly obvious to note the difference. In this case, the lower and middle frequencies are different: the lower frequencies increase, while the middle frequencies decrease. Thus, the parser correctly finds the transition at the 0.26 second mark.

Pan in a noisy environment

This particular case was picked because I knew it would give the system difficulties. Pans are difficult to capture using background noise levels as a cue, mainly because there has to be a significant change in levels from one place to the other. However, this particular pan was captured without any trouble. The answer can be found in that most camcorder microphones pick up approximately an angle of 40° in front of them. If the pan covers an area smaller than that described by the 40° angle, then the background noise characteristics will be exactly the same for both the beginning and end of the pan. Thus, the system captured a transition exactly after the 40° angle was swept. Again, the first level of the parser captured the transition point at the 6.4 seconds mark, which is exactly after the camera stopped moving. The three frames before and after are sent to the second parser, which evaluates the spectral characteristics. Figure 9 shows the spectral content before the transition.

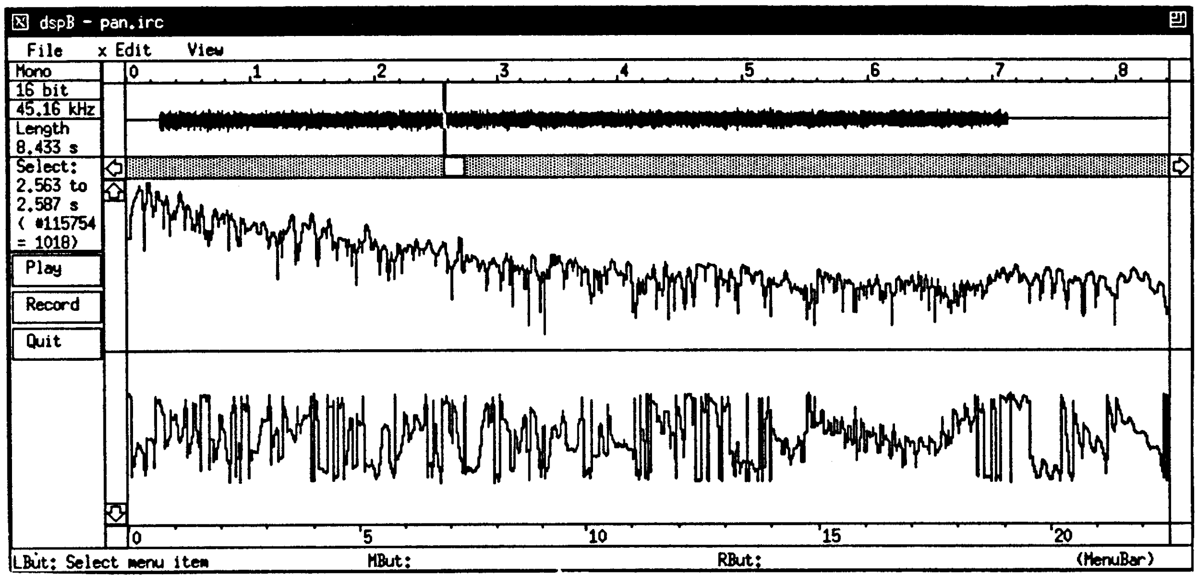


Figure 9: Pan in a noisy environment Before

Figure 10 shows the frequency content after the transition.

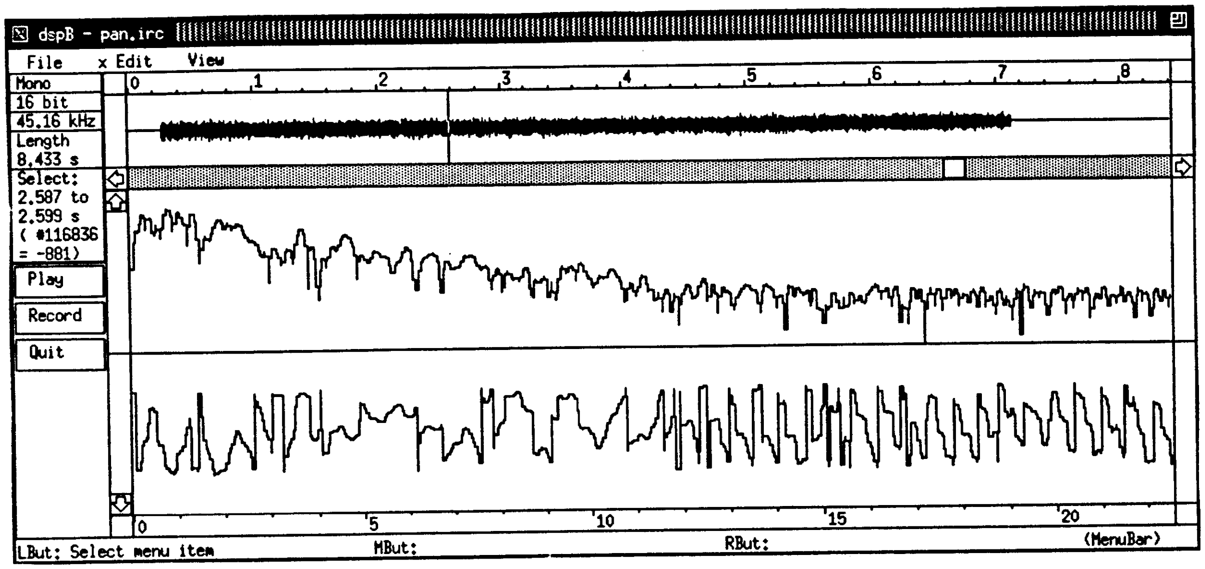


Figure 10: Pan in a noisy environment After

There is not much change in the frequency characteristics, so the second parser is not able to discern a change. However, the transition is still presented to the user, since the first parser found a significant change in amplitude.

<u>Music in the background</u>

Again, this example was picked because it would cause problems. There is music in the background both before and after the transition, and the music is at a low point in the second shot. The problem is that this second shot's signal level after the transition is within the tolerance level set by the first shot. So, the first parser only assumes a shot transition after the signal level of the second shot has surpassed the threshold. at about 7 seconds. The second parser looks at the frequency content again. Figure 11 shows the spectral content before the assumed transition.

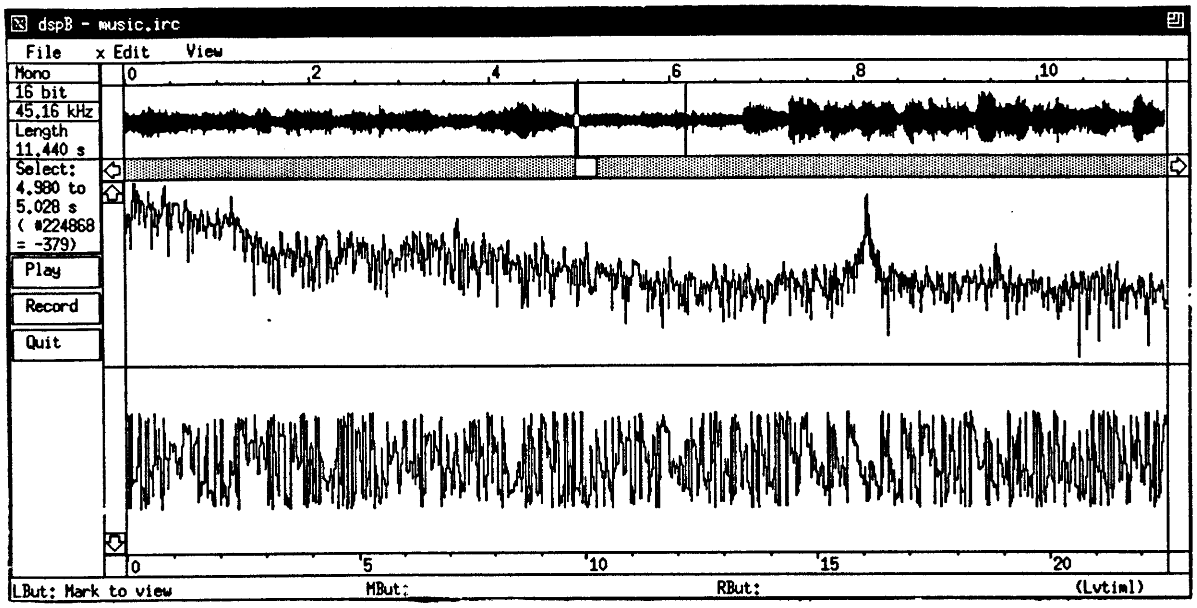


Figure 11: Music in the background- Before

Figure 12 looks at the frequency content after the assumed transition.

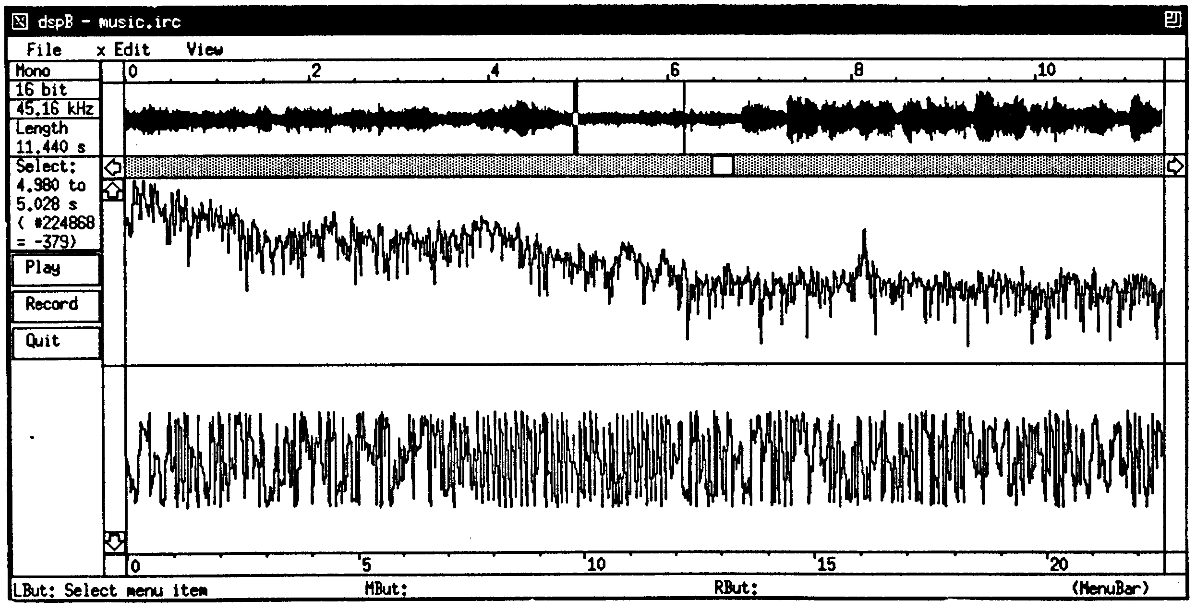


Figure 12: Music in the background- After

It is fairly obvious to note that there is not much change in the frequency content. Thus, the second parser reports no changes, and assumes the transition point chosen by the first parser, which is wrong. The transition found is actually about half a second after the real transition.

This mistake could be corrected by lowering the signal threshold. However, it was found that lowering this threshold would cause the parser to capture false changes every time there was speech present in the signal: most word transitions would be considered a shot change. Since most home movie material consists of speech, a compromise value was chosen for this threshold, eliminating errors caused by speech, but losing out on other transitions.

Shot change after car drives by

This is another challenging transition for the parser: there's speech present right after the shot change. However, since there was a car driving into scene, the threshold level was fairly high, since it is defined as a percentage of the signal content. Since the speech part falls within the tolerance, the first parser doesn't recognize it as a transition. Rather, it picks the point in which the speech utterance ends as a possible transition point, and sends the corresponding frames to the second parser for further analysis. Figure 13 shows the spectral content of the part before the supposed transition.

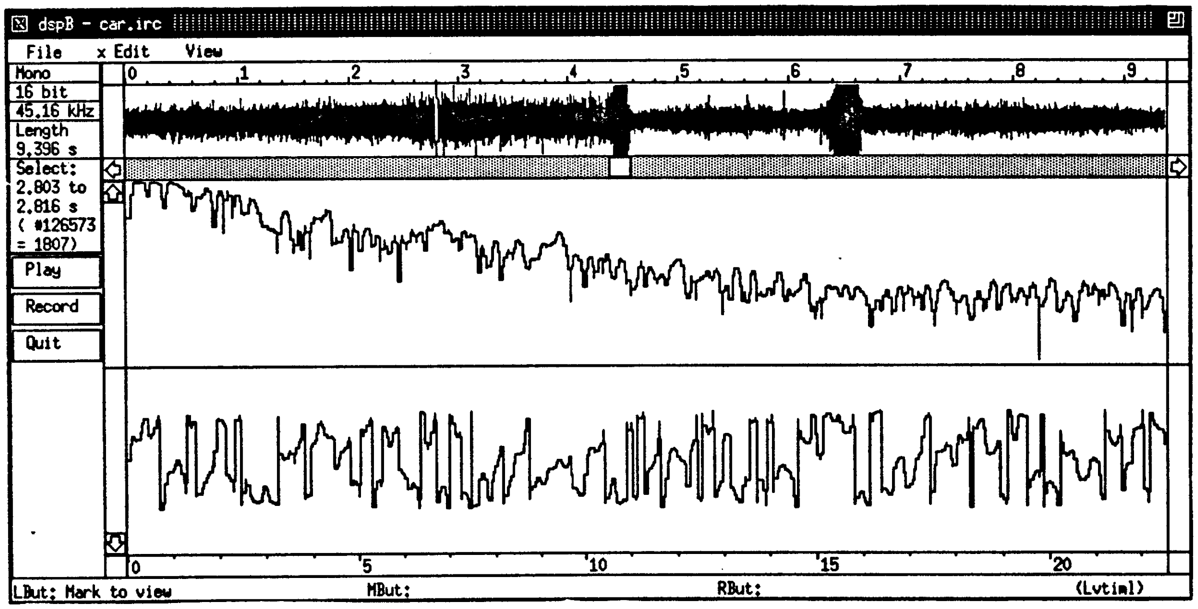
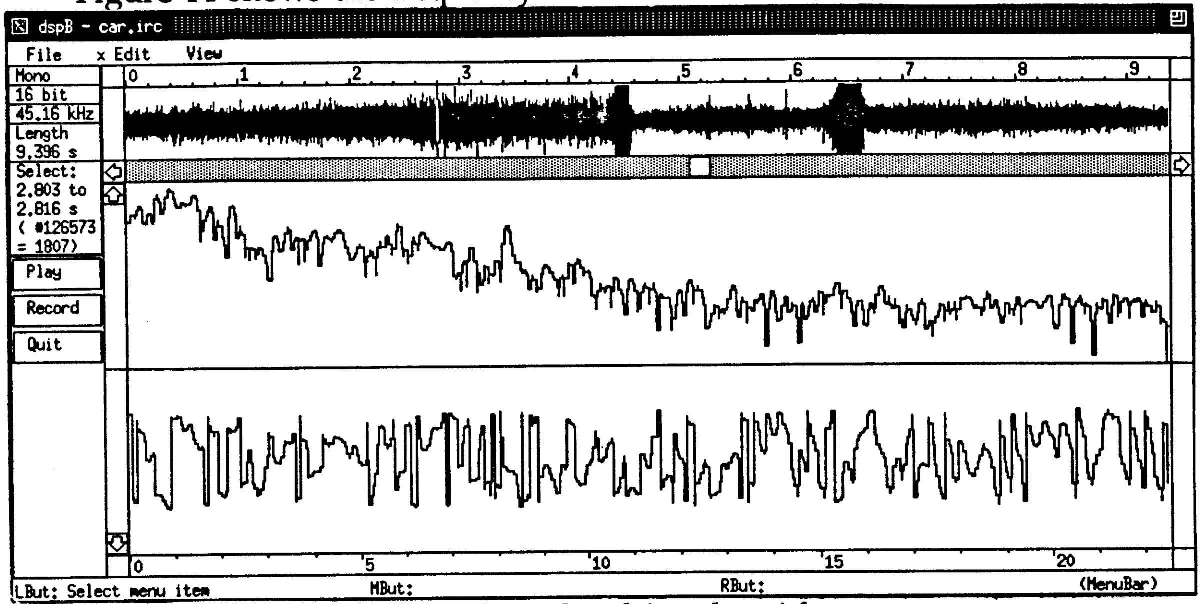


Figure 13: Car drives by- Before

Figure 14 shows the frequency content after the transition.



Mono
16 bit
45.16 kHz
Length
9.396 s
Select:
2.803 to
2.816 s
(#126573
= 1807)

Figure 14: Car drives by- After

Again, there is a significant change in frequency content, particularly in the lower frequency bands. Thus, the parser picks the point at 4.6 seconds, which is exactly after the speech utterance.

One more time, the setting of the threshold level determines a wrong transition point, since the real transition is before the utterance. However, an argument can be made that the transition is partially correct, since it is highly unlikely that a proper shot would begin with a partial speech utterance!

5. Observations

5.1 Limitations

There are certain limitations to the system. Some can be attributed to the original concept of parsing background noise levels, like detecting pans and zooms. Others are caused by design and implementation constraints.

Detection of a pan is quite difficult. In this case, it is only possible if the background noise conditions in the room change drastically from one position to the next. In other cases, it has been found that the parser will assume it is all in the same shot. The pan described in detail in the "System Performance Evaluation" section is a good example of this: the system only captured the transition after the pan had swept more than the 40° angle that the microphone is able to capture. The first, quick pan, was not captured.

This might be improved through the development of other heuristics, based on cues other than background noise.

Zooms are, by definition, impossible to capture using background noise parsing, since the room tone doesn't change at all. A zoom is a purely visual event, and thus it is impossible to find any identifiable representation on an audio track. The only way that zooms could be captured using audio exclusively would be to be able to extract the particular sound made by pressing the zoom button, if such a sound can be found.

Another limitation found during the evaluation of the system was that of the setting of threshold levels for shot changes. The parser uses a certain tolerance value, defined as a percentage of the overall signal level, to test discontinuities in sound levels. Those transitions found to be within the tolerance level are not considered shot changes, while those that exceed it are then transferred to the second level for further parsing. It is obvious to point out that, as this tolerance level changes, so will the number of transitions captured. A higher tolerance level helps in dealing with speech, which is apt to change in amplitude drastically; yet it results in a lower number of transitions captured, for it will only capture drastic changes in signal levels. A tradeoff was made in the parser design, in that an intermediate value was picked. The results of this tradeoff can be seen in the "Evaluation" section, in which the last example, that of the car, is not captured as a shot transition since the speech that is present at the beginning of the next shot is still within the tolerance level. Yet in the first example, even though the transition is from silence to a very low signal, the parser is able to identify it as a possible transition.

A third limitation found was that of setting the frame tolerance during parsing. As described earlier in the "Implementation" section, a frame tolerance of 43 was used. This frame tolerance is how long must a change in signal level be, without dropping below the current minimum, before it has to be considered a possible shot transition. As in the previous case with the setting of threshold levels, it is clear that changing this frame tolerance value affects the results. A longer tolerance might help capture longer events, while a shorter tolerance might miss some speech utterances, and take them for possible transition points.

The other important issue that this setting of a frame tolerance brings up is that of granularity: how long does a segment have to be before it is considered a shot? The value chosen is almost a second and a half long. Is that long enough to be considered a shot? As we break new ground with projects such as the one being described here, it is imperative for theoreticians and philosophers to begin redefining the concepts we need to rely on.

5.2 Sample Interaction

This section will describe a sample interaction between the user and the system. The interaction begins with the user digitizing the audio track of the video to analyze following the procedure described in the implementation section: first, the "Sound Designer" application must be opened. Recording of a soundfile begins by pulling down "New" from the file menu. An interface like the one shown here appears.

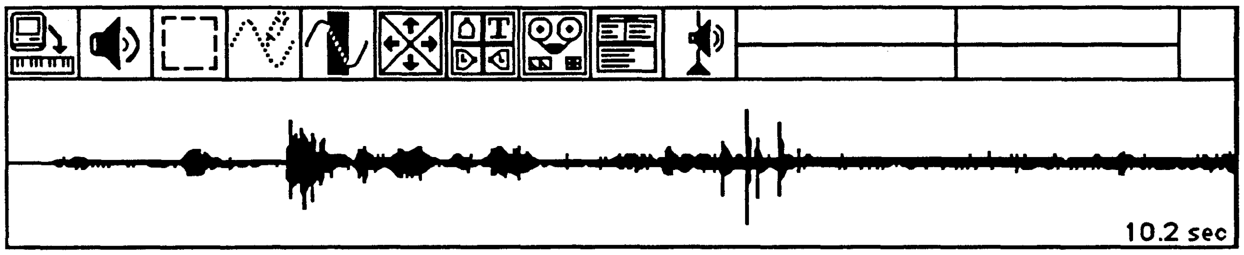


Figure 15: Sound Editor Interface

To begin recording, click the tape recorder icon. A standard Macintosh new file interface appears, with radio buttons at the bottom to select soundfile format. The soundfiles must be recorded with the "Sound Designer II Mono" option turned on, which is achieved by clicking on the appropriate radio button. This will record mono soundfiles with 16 bit resolution, which gives them excellent dynamic range and signal-to-noise ratio.

Once this is done, the user will open *Sancho,* the application. *Sancho* will proceed to prompt the user for the location of the digitized audio track, using a standard Macintosh "open file" dialog. Once the soundfile has been located, the parsing begins. The user is then shown a dialog which displays a scrolling bar showing how much of the first-level parsing has been completed.

If further parsing is required, the user will be informed of this by a separate dialog, which will also keep the user informed of the progress of the parsing.

Once the first possible shot has been found, another dialog appears. This dialog informs the user that a shot has been found, and ask if playback is desired.

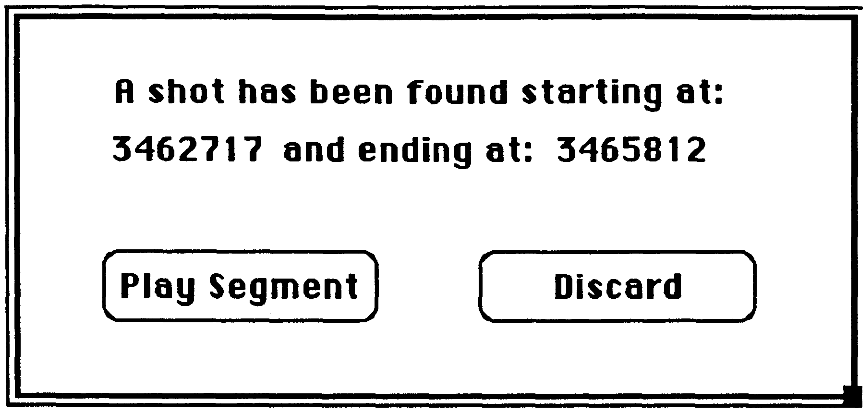

A shot has been found starting at:
3462717 and ending at: 3465812
Play Segment
Discard

Figure 16: Shot Found Dialog

The latter is selected by clicking on the "Play Segment" button on the dialog box. The shot is then played back to the user, who is then prompted by a dialog box on whether the shot is to her liking, or if there is a need to adjust the endpoints. The respective buttons are at the bottom of the dialog box.

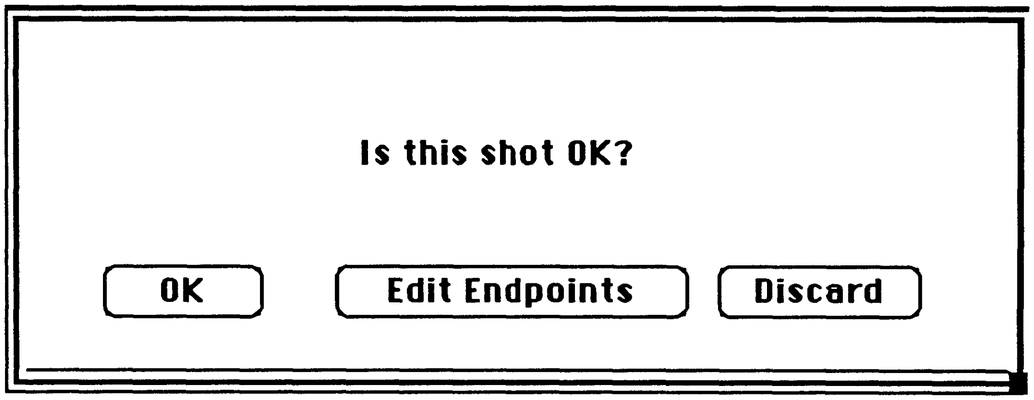


Figure 17: Shot OK Dialog

By clicking on the "OK", the user then proceeds to log the shot in another dialog pops.

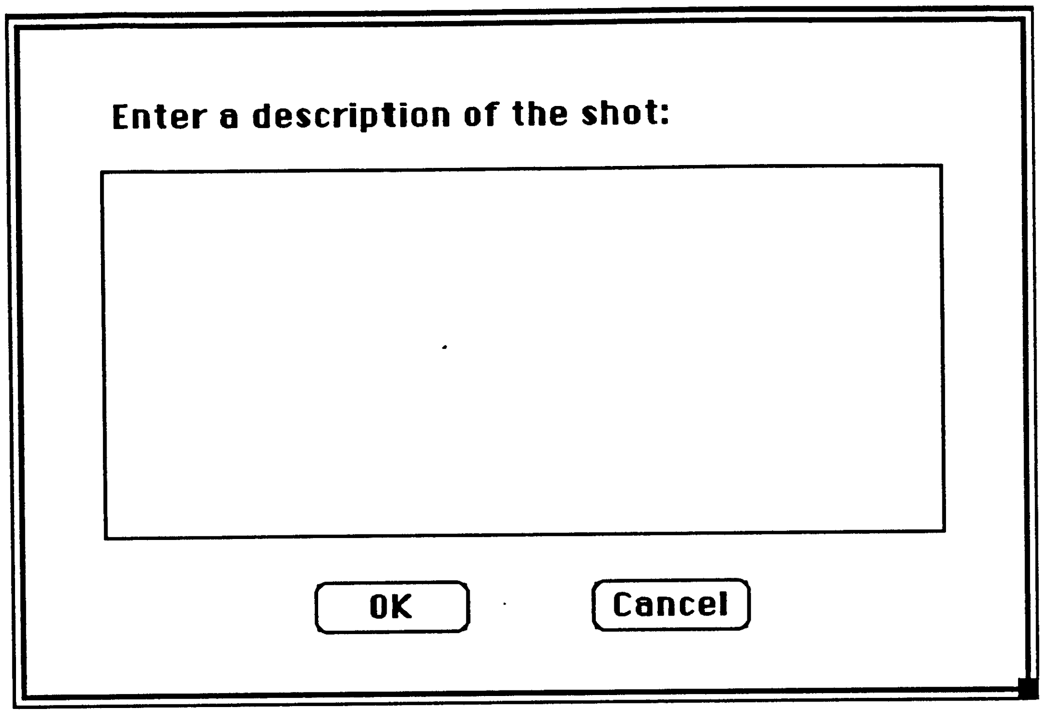


Figure 18: Shot Description Dialog

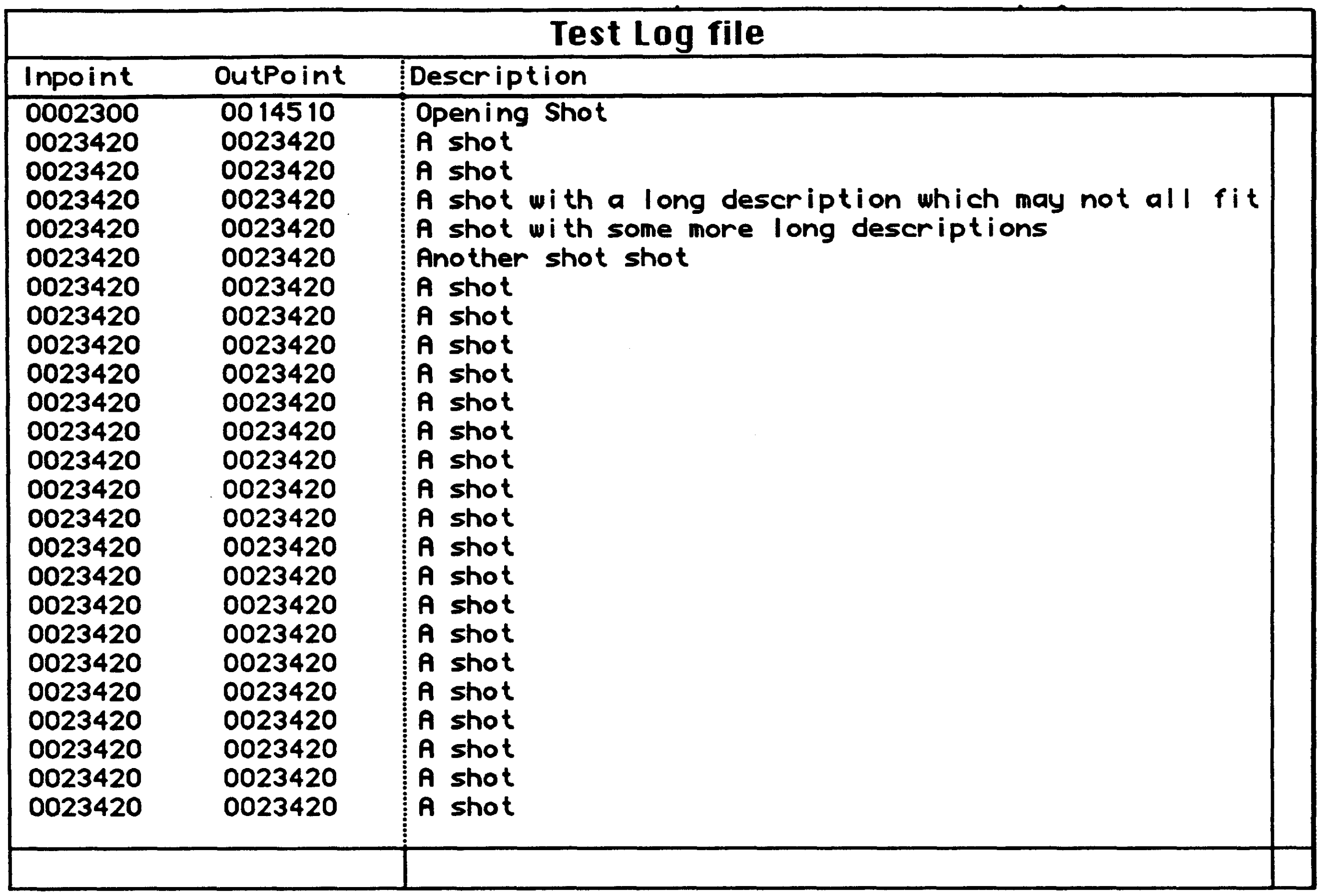
If the user clicks on the "Cancel" dialog, he's taken back to the last dialog. In fact, this is a constant throughout the interaction: anytime "Cancel" is clicked on, the prior dialog appears.

Shot endpoints and descriptions are stored on a log window, which is invisible to the user, except for two cases:

a) The user wishes to check on what has been logged. In this case, the log appears as a dialog box, with an "OK" button at the bottom.

b) After the logging process has been completed, the whole log comes up as a window, in which the user can again change the text description of the shot, or adjust the endpoints.

The log window looks like this:



Test Log file

Inpoint	OutPoint	Description
0002300	0014510	Opening Shot
0023420	0023420	A shot
0023420	0023420	A shot
0023420	0023420	A shot with a long description which may not all fit
0023420	0023420	A shot with some more long descriptions
0023420	0023420	Another shot shot
0023420	0023420	A shot
0023420	0023420	A shot
0023420	0023420	A shot
0023420	0023420	A shot
0023420	0023420	A shot
0023420	0023420	A shot
0023420	0023420	A shot
0023420	0023420	A shot
0023420	0023420	A shot
0023420	0023420	A shot
0023420	0023420	A shot
0023420	0023420	A shot
0023420	0023420	A shot
0023420	0023420	A shot
0023420	0023420	A shot
0023420	0023420	A shot
0023420	0023420	A shot
0023420	0023420	A shot
0023420	0023420	A shot

Figure 19 -Shot logger

As it can be seen, the shot log has shot endpoints clearly labeled, and offers a text-edit field in which the user can modify the shot description. Once the shot has been logged, the parser begins looking for the next shot, while displaying again the scrolling-bar dialog.

If the shot chosen by the parser is not to the user's liking, he can click on the "Edit Endpoints" button of the dialog shown in Figure 17, which will pop up another dialog that will let the user adjust the shot endpoints, as well as play the shot back as many times as the user needs in order to adjust the boundaries.

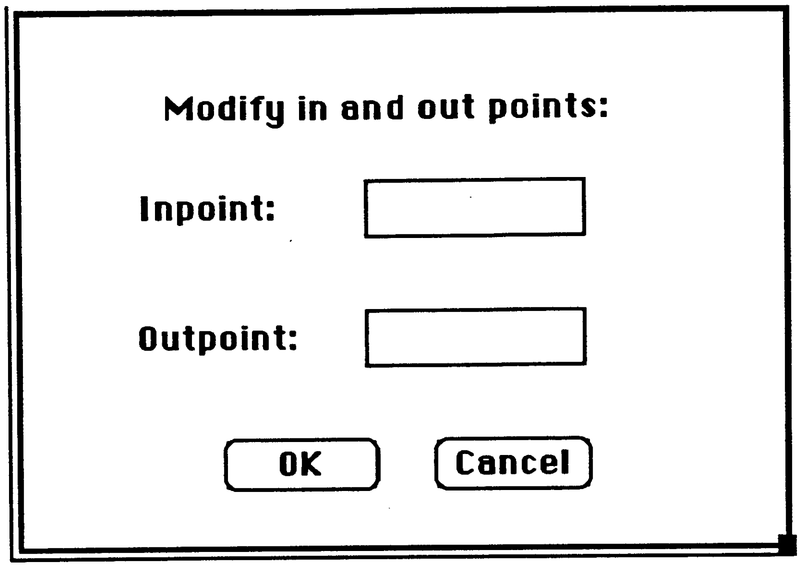


Figure 20: Edit Endpoints Dialog

Once the user finds the shot to his liking, he can get to the shot logger by clicking "OK". Else, he has an option to either scrap the piece, or send it back to the parser for further checking. This options are again presented in the form of a dialog box. If the user decides to scrap the piece, the parser begins anew from the endpoint of the scrapped shot. If the user clicks on "Check Again," the parser will begin searching again through the same material.

5.3 Directions for further research

There are several directions for further research. The following are the ones I think can provide greater flexibility to the system, as well as increase functionality. As computing power increases, other methods might be developed to improve system performance, using other, more-powerful parsing algorithms.The proposed directions for further research are real-time parsing, the use of spectral signatures, and incorporation of user profile/preferences.

5.3.1 Real-time parsing

Real-time parsing would increase functionality a great deal. This real-time addition would be implemented by running the audio output of the video player straight into a NuBus digitizer board (like the Digidesign "Sound Accelerator" card currently being used), and writing DSP 56000 microcode that would permit some time-domain analysis of the signal to be done as the tape is being played back. This would also require the writing of a library of routines to record and play back soundfiles, since the software being used now (Digidesign's "Sound Designer") does not allow for any other operations to occur while recording. By writing those functions, the recording of the audio track could be incorporated into the system, making it unnecessary to use the commercial software being used now. These changes would drastically reduce the amount of time necessary to do the parsing, since half of it will be done as the soundtrack is being digitized. They would also make the system more interactive, since response time will be considerably shortened.

5.3.2 Use of spectral signatures

This is an area of research that could bring exciting progress. Spectral signatures are defined to be the unique spectral content of a particular sound. Take, for example, the sound caused by an accidental tap on the microphone. This sound can be recorded, and analyzed. The spectral content of this noise can then be kept as a template for future reference, to check for possible occurrences in a soundfile. Pattern matching techniques from the field of speech recognition can then be used to compare the template with a random noise that is believed to be a tap on the microphone. Such techniques provide exhaustive methods for template matching, such as dynamic time warping (used for matching sounds of similar spectral content but different time duration). If a positive match occurs, the noise has been identified.

Such templates can be built for a number of different sounds which are usually encountered in most homes movies: cars, airplanes, etc. Thus, it will be possible for the system to not only recognize shot boundaries, but also to catalog some of the content of the shot as well. This will allow the system to create a log of the raw material. Also, this can lead to the implementation of a "search" function, that would be able to search through the material to find specific sounds. Suppose a user wants to find a shot in which there's a car. She would then call the search function, with "car" as an argument.The search function could then find the template of a car, previously recorded and analyzed, and run it through the material until a match is found. Then, the shot containing the car would be played back to the user. This would provide an initial shot description [AS 91].

Another direction such research could take is the creation of filters specifically designed to eliminate a particular sound. For example, take a plane flying overhead. Once it has been established that we have a plane flying overhead, using spectral signature matching, we could create an "airplane" filter, which would, once convolved with the original signal, eliminate the airplane noise from the soundtrack. This technique of eliminating sounds by subtraction is common in dialog processing, which is usually done in the analog domain, although some digital filtering is also beginning to be used.

In this technique also lies the key to extending the system to be used for commercial filmmaking. By creating a template for the clapboard sound, and the assistant director yelling "cut," it would be relatively easy to find all the takes, before proceeding to extract additional information from the audio track. This would reduce the amount of time necessary to do the initial logging of the raw material, and thus reduce production costs.

5.3.3 Incorporation of user profile/preferences

Adding a mechanism for storing user preferences would also improve system performance. Such mechanism could take two forms: a "preferences file" which the user might be able to create, in which he would select his options; or a system-created user profile, in which the system will would record a user's choices during interaction. Both options, or a combination, would provide much more flexibility, as well as improve the quality of the interaction. By using such a profile, the system would be able to recognize whether a particular user prefers longer shots, shorter shots, or tighter cuts. This would require some form of login, to identify the user to the system.

6. Conclusions

There are several conclusions to be drawn from this work. First and foremost, is the establishment of the concept of the Assistant Editor, and the usefulness of audio analysis in that concept. It is my firm belief that more people will edit their footage if they get useful help from their computer. It is not difficult to foresee the home entertainment center of the future, with a computer-controlled VCR with editing capabilities: the user logs the raw footage interacting with the computer, and later edits it based on the log that was created earlier.

Time constraints have prohibited the implementation of real-time parsing, as well as extensive subjective testing of users interacting with the Assistant Editor. Such tests should prove quite useful.

The work described herein has proven the viability of using sound as a content indicator. I have established that ambient sound offers clues about the physical and geographic aspects of place, since the microphone/sound recorder preserves the environment in which it is situated. Room tone is thus more than just ambiance: it becomes the specific attribute of the time and space in which the shooting is being done. Segmenting the video material becomes then the process of finding changes in the environmental factors that make up the room tone. It is viable to find most shot changes parsing audio exclusively. However, if complete segmentation is desired, it might be necessary to integrate this system with a video parser like the one built by Sasnett, which has already been described.

The use of spectral templates for feature extraction, although not completely implemented, shows promise as a vehicle for automating the logging process.

As stated in an earlier section, there seems to be a trend towards separate audio and video servers due to several reasons, like granularity of descriptions. The results of this work support that trend. This work has shown that the single frame, the unit of choice for video, is too coarse a description for audio. Sampling at CD quality (44.1 KHz), there are 1470 samples for every frame of video. It's fairly obvious that quite a few things can happen within so many samples. Thus, audio has a finer granularity than video. How small? For this project, the unit has been the FFT window size, of which there are several per video frame, depending on the number of bins in the FFT. Since this representation also varies, the single sample should be considered the unit of granularity.

This contextual-based representation helps in the integration of audio in Multimedia systems, as it ties in with context-based representations of the moving image like Stratification [AS 1991]. The Stratification methodology breaks down the environmental factors into distinct descriptive threads from which inferences concerning the content of each frame can be derived. Once a specific audio attribute is found, like a police car passing by, it becomes a part of the description of the shot being examined. The converse is also true: the shot description can provide information that, used with the "spectral signature" of the particular sound, can create new templates. For example, assume that we have a car driving through a beach. Parsing the audio track, we've found that the "car" template is present on this shot. We also know from the shot description that this particular car is on the beach. We can then use the audio corresponding to that particular segment as "car on the beach" and save it for fine-grained parsing at a later stage.

The work presented herein is only at its beginning, as there is still a great deal of ground to be covered. It was frustrating at times to see that the more progress was made, the more there was to do. There is a wealth of information encoded in the audio track, and more ideas and new analysis techniques are necessary to be able to parse such information. As computer capabilities increase in terms of speed and memory, it will be possible to obtain substantially greater amounts of content information, not just environmental factors and shot transitions. Speech recognition also offers some interesting possibilities if integrated into this system. Imagine having the computer do full transcriptions of a press conference, with shot changes, complete annotations, and full dialog transcription. It is my deepest hope that someone will continue working with the ideas and concepts I've tried to outline in these pages.

Bibliography

[AS 91]. Aguierre Smith, Thomas G. *Stratification: Toward a Computer Representation of the Moving Image* , working paper, Interactive Cinema Group, Media Laboratory, 1991.

[Applebaum 90]. Applebaum, Daniel. *The Galatea NetworkVideo Device Control System* , MIT, 1990.

[Arons 88]. Arons, Barry et al. *The VOX Audio Server* , IEEE Multimedia '89, Ottawa, Ontario, April 1989.

[Benson 88]. Benson, K. Blair. *Audio Engineering Handbook* , McGraw-Hill, New York, 1988.

[Bloch 88]. Bloch, Giles R. *From Concepts to Film Sequences.* Yale University, 1988.

[Chalsen 75]. Chalsen, Richard. *Cinema Naïveté: a study of home moviemaking as visual communication,* in Studies in the Anthropology of Visual Communication, vol. 2, 1975.

[DPAS 91]. Davenport, G., Pincever, N. and Aguierre Smith, T. *Cinematic primitives for Multimedia* , to appear in IEEE Computer Graphics and Applications, June 1991.

[Eis 49]. Eisenstein, Sergei. *Film Form.* Harcourt Brace Jovanovich, Orlando, 1949.

[Ellis 91]. Ellis, Dan. *Some software resources for digital audio in UNIX,* Music and Cognition Group technical document, MIT Media Lab, April 1991.

[Handel 1989]. Handel, Stephen. *Listening* , MIT Press, Cambridge, 1989.

[Laurel 90]. Laurel, Brenda. *The Art of Human-Computer Interface Design* , Addison Wesley, Reading, 1990.

[Lipton 75]. Lipton, Lenny. *The Super 8 Book* , Simon and Schuster, New York, 1975.

[LMKC 89]. Leffler, Samuel J. et al. *The Design and Implementation of the 4.3 BSD UNIX Operating System* , Addison Wesley, Reading, 1989.

[MC 85]. Mast, Gerald and Cohen, Marshall. *Film Theory and Criticism* , Oxford University Press, New York, 1985.

[Minsky 85]. Minsky, Marvin L. *Society of Mind* , Simon and Schuster, New York, 1985.

[Moorer 86]. Moorer, Andy. *The Soundroid* , videotape of the Media Lab Forum on 2/12/86.

[Nilsson 80]. Nilsson, Nils J. *Principles of Artificial Intelligence* , Tioga, Palo Alto, 1980.

[ND 86]. Norman, Donald A. and Draper, Stephen W. *User Centered System Design* . Lawrence Erlbaum Associates, New Jersey, 1986.

[OS 89]. Oppenheim, A. and Schafer, R. *Discrete-Time Signal Processing* , Prentice Hall, New Jersey, 1989.

[PA 84]. Pincus, Edward, and Asher, Steven. *The Filmmaker's Handbook* , Plume, New York, 1984.

[RS 78]. Rabiner, and Schafer, R. *Digital Processing of Speech Signals* , Prentice Hall, New Jersey, 1978.

[RM 78]. Reisz, Karel, and Millar, Gavin. *The Technique of Film Editing* , Focal, Boston, 1978.

[Rubin 89]. Rubin, Benjamin. *Constraint-Based Cinematic Editing,* MSVS Thesis, MIT, June 1989.

[Sasnett 86]. Sasnett, Russell. *Reconfigurable Video* , Master's Thesis, MIT, February 1986.

[WA 70]. Worth, Sol, and John Adair. "Navaho Filmmakers," *American Anthropologist,* volume 72, number 1, February 1970.