MULTIPLE VIEWS OF DIGITAL VIDEO

Eddie Elliott
MIT Media Laboratory
Interactive Cinema Group
March 23, 1992

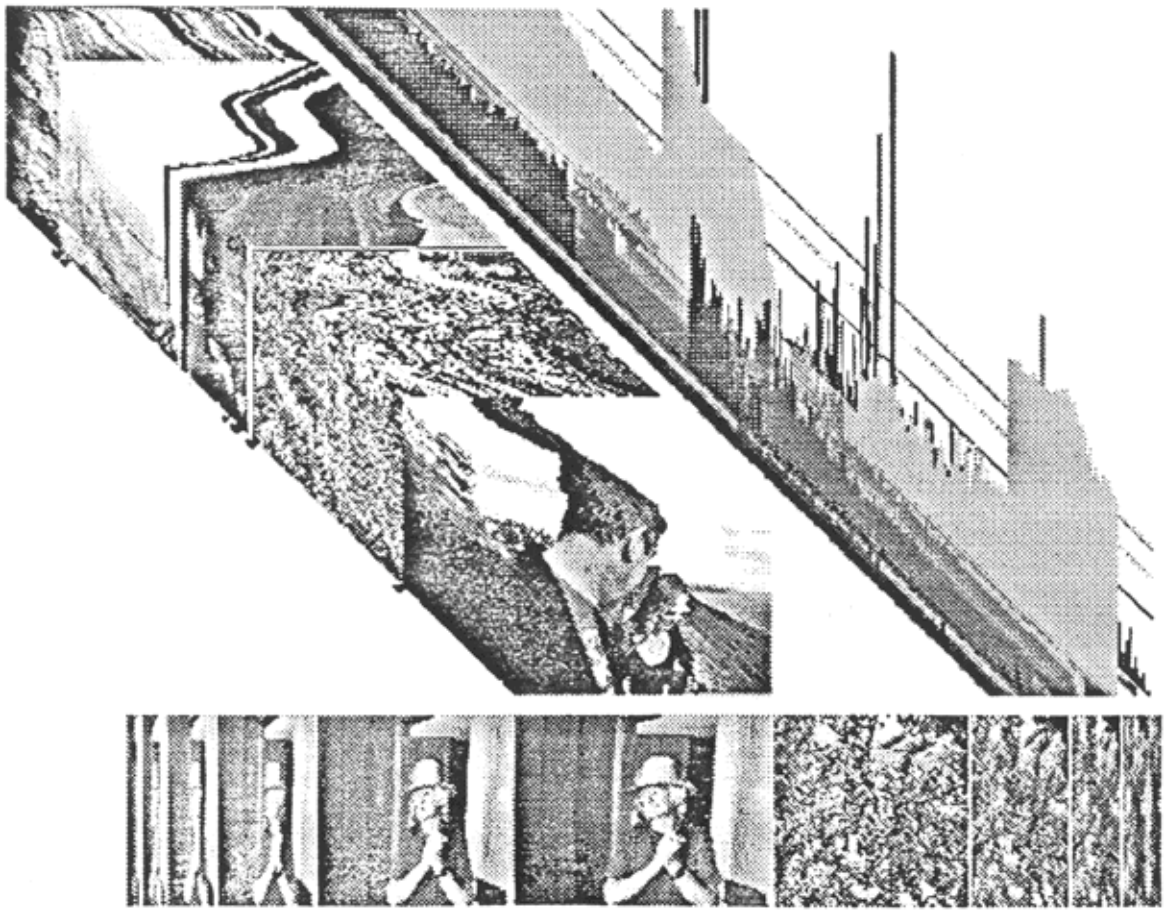
ABSTRACT

Recordings of moving pictures can be displayed in a variety of different ways to show what they hold. The historical and most absorbing way is to display the images through a rapid succession of full screen frames. However different forms of presentation can be used to emphasize different attributes. The video streamer positions frames of digital video sequentially in front of each other with a slight offset from frame to frame; visually this appears as a three dimensional extrusion of the video stream in time which emphasizes differences along the side and top edges of adjacent frames. In this way the video streamer helps us see characteristics between frames and across shots such as transition types and cutting rhythms. While viewing the video stream one can select bounds of interest in time; this area can be adjusted using a rubbing motion along the stream. The micro-viewer shows us more precise frame to frame relationships, based on the portion of the video stream we have currently selected. The shot parser uses a frame differencing algorithm to offer a helpful element of machine assisted abstract analysis.

These three components, the video streamer, the micro- viewer, and the shot parser, are described in terms of functionality and their relationships to one another. An architecture for a more extensive parsing algorithm is suggested. And some possible applications for these tools are also discussed.

INTRODUCTION

Our video streamer and its accompanying micro-viewer and shot parser provide a small tool set for analyzing and appreciating a stream of moving pictures in various ways. This tool set can combine with other tools and applications to present multiple perspectives of video streams. Because it presents itself both dynamically and statically, and because one can view the stream as is or dive in and manipulate it, the video streamer offers a series of conceptual bridges from the concrete experience of watching moving pictures, to a slightly removed perspective (from a view of the frame to a view of the stream), to an entirely abstract analysis of the signal as it might relate to features of the stream's content These separate views are synchronized and displayed simultaneously, providing a coherency that helps bridge concrete and abstract appreciations of the stream.



THE VIDEO STREAMER

Digital video permits new ways of presenting a video stream. Moving pictures are normally presented in a rapid succession of still frames that produce the illusion of motion. We might view this string of pictures as a stream flowing past us in time. When we stop the stream and step back from it to gain a perspective beyond a single frame we usually view strings of frames as though they were still on celluloid strips. The video streamer offers two alternatives to these types of views. First, it stacks the frames flip book style so that we can view more of the stream in the same space. lining up the frames so that we see only their edges, we can see temporal characteristics that are perceived in the normal full-screen full-motion view but aren't visible in the film-strip view. Characteristics such as camera motions, transition types and locations, shot length, and shot rhythm stand out. Second, since the video stream is inherently dynamic, the extruded view also presents itself as a flowing stream. The edge patterns in the extruded view give clues to the characteristics of the stream's contents, but they rarely indicate exactly what is found within the frame. As we view the stream flowing. the images are implanted in our minds. Then, when the extruded stream is paused, its edges provide reminders to what we've seen, as well as a view of temporal attributes. Maybe this reminder characteristic of the extruded stream caters to our spatial memory. It's sometimes easier to remember where something is than when it is.

(some related questions: *How long do viewers remember the exact sequence of shots? How rapidly and in what fashion does the ordering deteriorate in the*

viewer's mind? How much longer can viewers catalogue content than they can list its sequence? How does content and its presentation relate to these memories?)

Our current implementation portrays the video stream as a solid opaque block. If the computer can separate foreground from background, or static elements from moving elements, we might also employ transparency in the rendering of the extrusion to selectively view certain elements. For example, we might render the background portion of each frame transparent so that we only extrude foreground elements. This might be a way of seeing the contents within a block.

As the video streamer stacks incoming frames, it flows from front to back with the farthest frames representing the portion of the stream that is downstream in time.

When the stream is paused, the extrusion invites poking at it to review' its contents. As you move the pointer across the extrusion, the frame immediately below it is hi lighted and presented in full in a viewer below. By "stroking" the extrusion from back to front the user essentially replays a portion of the stream. Stroking front to back replays it in reverse. This provides an intuitive way of indexing the contents of the stream and to quickly jump from a fast-forward-like overview to a meticulous frame-by-frame review. It also relates the longer term characteristics visible on the sides of the extrusion to the contents of the individual frames that comprise the extrusion.

THE MICRO-VIEWER

The video streamer shows characteristics across hundreds of frames in the stream, Sometimes it is useful to look at things on a smaller scale. The micro- viewer presents a series of frames side by side with the center frame representing a user selected moment in the video stream, in our case the frame lying under the mouse pointer. The frames preceding and following this selected frame in the stream are presented to the left and right, but are more and more foreshortened the further they are in time from the central frame in a fish- eye fashion. This allows us to show' a longer string of frames within a given space. The foreshortened frames in this anamorphic view have just enough resolution to allow us to spot a cut coming along.

Moving the pointer across the stream extrusion scrolls the frames in the micro-viewer. This dynamic relation helps connect the global characteristics visible in the extrusion with the local characteristics visible in the micro-viewer .

SHOT PARSING AND THE HISTOGRAM DISPLAY

Sliced bread is handy for making sandwiches or toast. Our shot parser attempts to slice the video stream at shot boundaries, saving the viewer the tedium of finding precise beginnings and endings of shots. It compares successive frames, or series of frames, watching for gross changes in the signal, tagging suspected shot boundaries. The shot parser currently works best at recognizing cuts between shots. A more elaborate video stream parser could conceivably watch for signal characteristics that signify other types of transitions, dissolves, fades, wipes, and

keys for example. It might be tuned to recognize camera motions as well, say pans and zooms. [1], [2]

Each shot in a video stream has its own signal signature. Just like a written signature, a shot's signature has various characteristics that we can analyze to distinguish it from other shots. Our shot parser works in tandem with a video capture utility, so there are a number of aspects of the algorithm that we've intentionally kept simple so as to minimize the parser's impact on CPU load. In other words, we are looking at an especially blatant characteristic of the video signal's signature that costs little CPU time to analyze.

The shot parser currently samples each frame along its edges for RGB values and builds a color frequency histogram for that frame. We are sampling only along the top and left edges of the picture. We need at least one vertical and one horizontal sample region in order to filter out the possibility for pans and jibs as scene changes. We sample at the edge since in most shots the background is relatively stable and appears at the edges, while much of the action appears within the frame. A more sophisticated sampler might convert to another color space, say HSV or YIQ, but we are currently sticking with RGB frequencies to minimize processing. In our current system, we just compare the color histograms of a pair of frames and sum the absolute differences between the histograms at each color value. This summed difference may continually ride at a relatively high value for some footage, especially if the signal is noisy. So we take the difference between subsequent difference values to arrive at a plot that has fairly well defined spikes. When the spikes rise above a magic number threshold the given frame is marked as a transition point.

Our small sample and simple analysis allow us to maintain real-time capture simultaneously, but it does slip occasionally. It correctly ignores pans as potential scene changes, but it usually ignores dissolves and fades as well. Soap operas fool it frequently. With their frequent intercutting of close-ups the background seems to always be a dark wood tone that varies little across shots. Black and white movies also fool our current system since their changes in luminance from shot to shot are much smaller than the combined changes in luminance and chrominance the shot parser recognizes in color pictures.

The histogram can be displayed concurrently in an extruded fashion similar to the display of the video stream. Imagine a stack of vectorscope graphs.

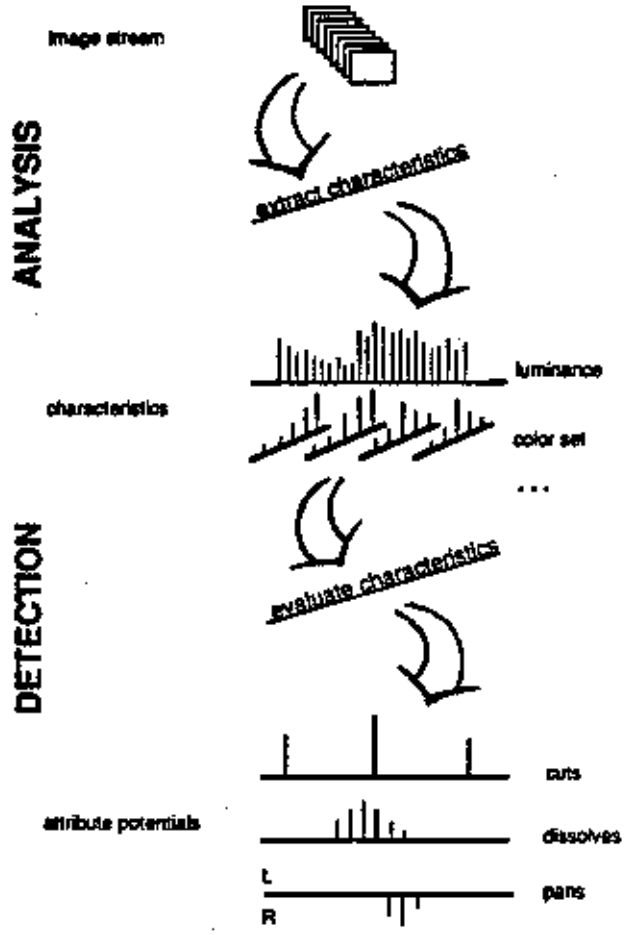
PARSING YET TO COME

There are many types of transitions to recognize in a stream beyond just shot boundaries. Shot boundaries do give us a useful first slicing. But how would we want to partition Hitchcock's "Rope" into meaningful chunks for example? We are constructing an architecture for combining more elaborate analysis and interpretation algorithms in order to tease more information from the stream with computer assistance.

We have implemented our shot parser, which currently only detects cuts and very fast dissolves, within a model for a more extensive parsing architecture. This

architecture separates parsing into two phases: analysis and detection. Analysis extracts characteristics from the image stream's signal. Detection evaluates these characteristics across time to derive potentials for various attributes of the image stream. Some examples of signal characteristics are luminance values, color set histograms. and noise levels. Some valuable attribute potentials might be cut locations, location and rate of dissolves, or direction and rate of zooms and pans. Our current parser only extracts the color histogram for a portion of each frame and derives the likelihood that that frame Is at a transition between shots. Both analysis and detection may go through multiple iterations.

Our shot parser first extracts the color histogram characteristic for a frame, then compares it with the same characteristic from the previous frame to get a color histogram differential characteristic, then compares these differentials from frame to frame to get a second differential characteristic that is finally evaluated against a threshold for detecting probable cut frames. The second differential is a cheap way of filtering out some false cut detections caused by noise in the signal.



A more sophisticated parser that incorporated a variety of characteristics would need to employ some weighting algorithm to arrive at a difference value, or perhaps a collection of values that translated the various signal characteristics into potentials for content attributes, fade to black or pan left for example. This

weighting may call for a mixture of some standard statistical analysis and some AI guessing. The accompanying audio stream also may provide clues to the overall comparator. [3], [4]

With the luxury of more CPU cycles we might look at luminance characteristics separate from chroma. We'd also want to filter out as much signal noise as possible. We might also perform Fourier transforms to compare frequency characteristics across time. A very simple characteristic that comes for free in some systems is a given frame's temporal compression factor. If some means of frame-difference compression is employed. the compressor commonly returns a compression factor for each frame it processes. This could be construed as a difference value between successive frames.

The nature of the sample has a strong influence on the characteristics produced by various signal analyzers. Some possible variations for the location of the sample area:

* along the edges
* in the middle
* the whole raster
* equally spaced points
* random points

For any given area, the sample size may vary from minimal to extra redundant. One might also average regions of pixels around a sample point to represent the sample for that point. And these regions may have various weighting strategies for how each pixel contributes to the final sample value at the point. Of course, we need to take into account temporal characteristics of the sample as well. We might filter a given sample point across time to eliminate some noise. And sample frequency will definitely impact the rest of the process. Depending on the algorithm, dissolves might be tagged as shot boundaries when we have infrequent samples but pass undetected when we sample every frame. For some characteristics it may be useful to vary the sample region over time. Perhaps the nature of the sample region could be defined dynamically by the analysis stage.

Our difference threshold has been derived empirically and is set relatively low in order to catch false hits rather than miss border-line differences. The threshold might also vary in a dynamic fashion determined by the analysis algorithm as was suggested above for determining the sample region dynamically. Another method for deriving thresholds is to train the system. By feeding the algorithm frame pairs or sequences of frames that a viewer recognizes as similar or different, the system can obtain its values and adjust it's interpretation of those values to correspond to the similar/different label provided by the trainer. Given a parsing algorithm, such a "training" session might yield a maximum delta value below which a pair of frames will be deemed similar and a minimum delta value above which frame will be deemed different. Hopefully the highest similarity value is below the lowest difference value. If so, then there is a range of ambiguity where we need to decide whether to include potentially false differences or discard slight differences. Otherwise, this range of ambiguity will extend into the high and low delta ranges the system has determined, rendering them not so definite.

APPLICATIONS

Potential applications for these ways of viewing video streams fall into at least three general realms: viewing, making, and analysis. Any particular activity may involve more than one of these. For example, editing involves both reviewing raw material and manipulating it to create more refined material.

In situations where one is viewing moving pictures the video streamer and its accompanying tools can be useful for reviewing and for getting a handle on things that change over time. Such situations are encountered while viewing hypermedia documents, while logging footage prior to editing or for entry into video databases, and during film analysis

When constructing documents that include moving pictures the streamer, micro-viewer, and shot parser are especially helpful with the coherence they provide between macro views of the stream, micro views of the content, and abstract views of the signal.

Current video engineering diagnostic and analysis tools are typically analog and look at a single frame at a time. Digital renderings of signal characteristics such as our streaming color histogram can help to make temporal aspects of the signal more apparent. And displaying this data in tandem with the picture stream helps one to correlate the concrete frame with the abstract rendering of its signal.

Working with digital video on personal computers can be a creative and stimulating process. But it's usually preceded by a tedious gathering stage that involves selecting material and then introducing it to the computer by digitizing it. This presumes that you know ahead of time what you ought to select. When the video streamer is used in conjunction with other tools and applications it is especially useful as a video digitizing utility. Capture utilities commonly allow you to digitize a "live" image and then trim it afterwards to meet your needs. If your image source is recorded, on tape or disc, you typically review your footage, cue up the source, trigger the source and the capture utility to do their things, and then trim the results. Reversing the order of capture and selection, the video streamer eliminates some of these steps. The video streamer has a rolling digital image buffer that works sort of like the seven second delay used on talk radio programs. By the time you view something for the first time, it is already digitized. You only need to select the shot you're interested in from the buffer. There is no synchronizing of tape and capture, and there is no trimming afterwards.

At some time to come our video source will arrive digitally and the benefits of digital vs. analog buffering will be mute. Regardless, the view provided by the video streamer is useful in the way it relates to a viewer's short term memory. (see the italicized questions above)

References

These both describe methods for detecting pans and zooms:

(1) Teodosio, Laura. "Salient Stills. " Master's thesis, MIT Media Lab, June 1992.

(2) Veda, Hirotada, Takafumi Miyuatake, and Satushi Yoshizawa. "Impact: An Interactive Natural-Motion- Picture Dedicated Multimedia Authoring System." CHI '91 Conference Proceedings, pp. 343-350.

These two suggest ways of segmenting and describing an audio stream :

(3) Pincever, Natalio. "1f you could see what I hear: Editing assistance through cinematic parsing." Master's thesis, MIT Media Lab, June 1991.

(4) Hindus, Debby. "Semi-Structured Capture and Display of Telephone Conversations. " Master's thesis, MIT Media Lab, February 1992.