A Semi-automatic Approach to Home Video Editing

Andreas Girgensohn, John Boreczky, Patrick Chiu, John Doherty,
Jonathan Foote, Gene Golovchinsky, Shingo Uchihashi, and Lynn Wilcox
FX Palo Alto Laboratory
3400 Hillview Avenue
Palo Alto, CA 94304, USA
{andreasg, johnb, chiu, doherty, foote, gene, shingo, wilcox}@pal.xerox.com

ABSTRACT

Hitchcock is a system that allows users to easily create custom videos from raw video shot with a standard video camera. In contrast to other video editing systems, Hitchcock uses automatic analysis to determine the suitability of portions of the raw video. Unsuitable video typically has fast or erratic camera motion. Hitchcock first analyzes video to identify the type and amount of camera motion: fast pan, slow zoom, etc. Based on this analysis, a numerical "unsuitability" score is computed for each frame of the video. Combined with standard editing rules, this score is used to identify clips for inclusion in the final video and to select their start and end points. To create a custom video, the user drags keyframes corresponding to the desired clips into a storyboard. Users can lengthen or shorten the clip without specifying the start and end frames explicitly. Clip lengths are balanced automatically using a spring-based algorithm.

Keywords: video editing, video analysis, video exploration, automatic video clip extraction.

INTRODUCTION

Video cameras are becoming increasingly popular for both home and office use. Video is commonly used to document family events such as vacations, weddings, and graduation ceremonies. In the office, video cameras are used to record presentations, user studies and field work, and are often taken on business trips to record people, places, and activities.

Furthermore, Digital Video (DV), a digital camcorder format, is becoming increasingly common. More and more personal computers are able to interface with DV cameras and have enough disk space to hold reasonable amounts of video. We believe that the combination of high-quality digital video, cheap capture cards, low cost disk space, and interest in creating video content for the Web will increase the demand for editors that handle non-professional video material.

It is difficult, however, to use this video after it has been recorded. While people may view such videos once or twice, they are typically left "in the bag," since the interesting parts are intermixed with longer, less interesting regions. Further, such video is often of poor quality resulting from abrupt camera movement or too short or too long views, making it uninteresting to watch while waiting for the next interesting section. Although tools exist for editing home video, most casual users find it difficult to use applications such as Adobe Premiere [1] or Apple's iMovie [2].

UIST '00. San Diego, CA USA

In this paper, we describe a new approach for addressing the problems non-professionals have in using existing video editing tools. Our approach is to provide the user with an interactive system for composing video that does not require manual selection of the start and end points for each video clip. The system analyzes the video to identify suitable clips, and a set of editing rules is applied to select the initial length of each clip. The user can then select clips, adjust their lengths, and determine the order in which they will be shown.

This process may be divided into several steps: The video is first analyzed to determine camera motion and speed. Bad video is typically characterized by fast or erratic camera motion. Motion analysis is used to find segments of the video, or clips, that are suitable for inclusion in the final video. A keyframe for each suitable clip is displayed to the user. We provide an interface that allows the user to browse quickly through the keyframes of the entire raw video. The user then selects the desired keyframes and organizes them in a storyboard.

Hitchcock creates the final video automatically by concatenating the selected clips. Editing rules are used to optimize the length of each clip to be included. These rules represent the experience of a professional video producer and embody heuristics about clip duration and transitions. After reviewing the automatically generated video, the user can use the storyboard interface to lengthen or shorten the individual clips in cases where the rules yielded unwanted results.

We conducted a user study in which we gave DV cameras to the participants to shoot some home video and had them edit the video with Hitchcock. The study was completed very recently so that only a few preliminary results are reported in this paper.

In the next section, we discuss designs of video editing systems that use different levels of automation. After that, we describe our approach for extracting clips from a video. We then present a user interface that supports semi-automatic video editing: the system offers a collection of clips that the user can select, adjust in length, and place in a desired order. We conclude with a discussion of the use of a spring-based

algorithm for balancing the lengths of the output video clips.

LEVELS OF AUTOMATION FOR VIDEO EDITING

There are aspects of the video editing process — selecting appropriate clips, for example — that are best performed by users. Other tasks, because of their tedious nature (e.g., selecting precise in and out points), are best done by the computer. Commercial and research video editing and abstracting tools, however, differ greatly in the amount and nature of the automation and manual intervention used. These systems can be classified by the amount of user interaction required to produce a final output. There are three natural groupings based on this measure.

Mostly Manual

There are many video editing systems that provide the user full control over the editing process, ranging from those made for professional video editors to newer ones designed for home users. None of these systems, however, performs video analysis to support the editing process. Most systems offer no assistance for selecting in and out points. Thus the user must examine the video on a frame-by-frame basis. Some home-user systems that support DV create an initial take list based on the camera on/off points. However, users need to trim undesirable material from the takes manually.

For amateurs, editing tools such as Adobe Premiere [1] or In-Sync's Speed Razor [11] are difficult to use, but with effort can produce professional results. Simpler editing tools, such as Apple's iMovie [2] and JavuNetwork [12] make it easy to select video takes and to arrange them in a storyboard. Trimming the takes, however, requires finding the in and out points for cutting. This is difficult even with the simpler tools because video must be examined on a frame-by-frame basis and only uniform sub-sampling is provided. The effort required ensures that only important videos are worth editing.

Fully Automatic

The other end of the spectrum removes all control from the user. The CMU Informedia system [5, 18] creates video skims that excerpt portions of a video based on text captions and scene segmentation. In this system, all video is production quality. Excerpts are selected automatically by analyzing the associated close-caption text and finding important passages. He et al. [10] create summaries of on-line training video using associated presentation material (PowerPoint slides), audio emphasis, and statistics from prior viewing. Pfeiffer et al. [16] find interesting locations in movies using audio analysis (e.g., finding gunshots). The MadCap project [17] produces summaries of videotaped Xerox PARC forums using presentation material and user indexing. All of these systems automatically produce a final result that is not easily changed by the user.

Semi-Automatic

In the middle ground are systems that automate some of the editing tasks in an attempt to make it possible to work with a larger class of video material.

The Intel home video abstracting system [15] allows a user to select an input video and a desired duration. The system chooses a small clip from each camera take based on the audio energy. The system then creates a four-level cluster tree of the video clips based on the recorded date and time. It randomly removes subtrees at different levels to reach the desired output video length. Based on the shooting time difference between adjacent clips, different video transitions are used. This system produces a somewhat random abstract that provides even coverage of the source material and emphasizes clips with high audio levels. The only user input is the total duration, and thus no control over which clips appear in the output is possible, but may depend on the genre of the video.

The Hitchcock system described in this paper allows a user to specify the desired total length of the output video and the lengths of the individual clips. The system presents the video clips that are of sufficient quality and allows the user to choose those to be included in the output. Hitchcock analyzes video to classify the frames of the video, and then determines video in and out points based on the suitability of the analyzed video material and the requested clip length. The users have as much control over the editing process as they want, but the tedious tasks of finding usable video clips and selecting in and out points are performed automatically.

FINDING VIDEO CLIPS

Hitchcock supports the video editing process by automatically detecting suitable clips in the raw video material and by presenting them to the user for selection and adjustment. To do so, it uses a number of heuristic editing rules to find good clip boundaries. These editing rules are encapsulated in a computational model that supports shortening and lengthening of clips without requiring the user to specify explicit in and out points.

Editing Rules

Experienced video editors have rules that reflect their editing decisions. We formalized these rules based on the experiences of one of the authors who is a professional video editor. These rules not only aid in the selection of in and out points, but also embody contextual information. We have identified a number of general rules for automatic editing in Hitchcock:

- A clip should be neither too long nor too short.
- Clips with too fast camera movement should not be included because they are difficult to watch and non-informative.
- A clip should have a minimum degree of brightness.

These rules guide our selection of appropriate algorithms for extracting clips from video takes. Additional or different rules can be accommodated easily with our approach.

Unsuitability Score

We implemented some of the rules as a simple but still expandable computational model that identifies useful clips in video takes. The model incorporates brightness and camera and object motion information extracted from the video, and can handle user requests to shorten or lengthen particular clips without requiring the user to specify in and out points explicitly.

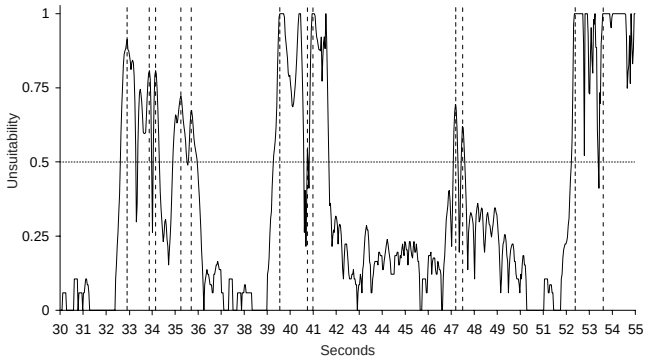

Figure 1: Unsuitability Score with Clip Boundaries

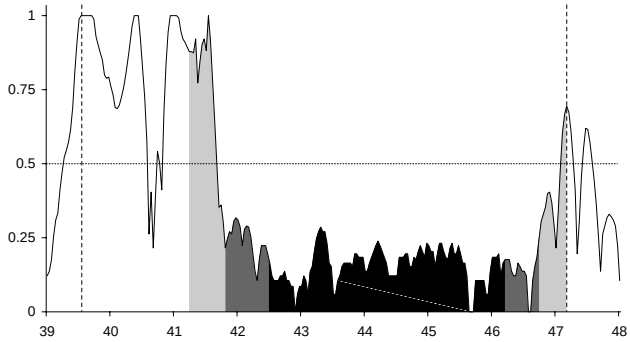

Figure 2: Selected Clip Portions

As described above, camera motion is an important criterion for determining clip boundaries. The camera motion needs to be determined independently for each take. Takes are based on on-off time stamps provided by the camera; when dealing with digitized analog video, camera changes can be approximated by comparing subsequent frames in the video. We determine camera pan by computing the direction corresponding to minimum RMS difference between adjacent frames. The first difference is computed based on a shift of 32 pixels; the second based on 16; etc. After each step, we continue in the direction that had the lowest RMS error in the pixel comparison. We found that using a sliding window of five video frames ($^1/_6$ second) for adding the detected motion vectors led to the best results for classifying camera motion into meaningful categories. We are currently working on a new approach that will detect camera zooms better.

We represent camera motion as a component of a single unsuitability score for each video frame. To determine a good mapping from the detected camera motion to a single number, we examined video material from different sources and classified the amount of camera motion into the subjective categories of good, acceptable, tolerable, or unacceptable. Using those observations, we grouped the amount of horizontal and vertical pan and the amount of zoom into several classes corresponding to those judgments. Vertical pan had a stronger influence than horizontal pan on the perceived unsuitability of a sequence of frames. We therefore use a weighted average of horizontal and vertical pan with more weight for the vertical component to represent those factors. We set the range of the unsuitability score between zero and one and divide that range evenly into the four categories described above. For mapping the categorized averages into the corresponding unsuitability values, the square root function produces the best fit.

For determining unsuitability based on the brightness of a video frame, we compute the fraction of the total pixels above a brightness threshold. For the videos in our library, we found that 20% of the pixels with at least 45% brightness is sufficient for a suitable frame. For the unsuitability score, we map the range between 0% and 20% linearly to a decreasing unsuitability. We combine the two unsuitability measures by taking the maximum.

Figure 1 shows a chart of a typical unsuitability score. In addition to the brightness and camera motion discussed above, other factors can contribute to the unsuitability score. This approach is not limited to the visual domain. For example, audio features could also be used to determine good portions of a video clip. By including all these factors in a single score, the clip segmentation and trimming can be performed independently of the set of features used.

Determining Clip Boundaries

Most commercial systems for editing DV extract information about takes from the video and visualize each take with a keyframe. It is often beneficial to further subdivide takes into clips based on camera motion and other factors. For example, a take might contain two still portions separated by a fast pan. In such a situation, it is better to have two clips and to trim each of them independently rather than attempting to manipulate the whole take. For the material shot by the participants of a recently concluded user study, our system segmented video takes into 2.8 clips on average.

We use the unsuitability score over the duration of a video take to find one or more suitable clips with respect to our criteria. By selecting clips that fall into “valleys” between peaks of the unsuitability score, we have a portion of low unsuitability that can be expanded into the higher areas if needed. At the same time, these clips must meet the minimum length requirement. We start by setting an unsuitability threshold and then finding the peak for each part of the unsuitability curve that is completely above the threshold. Those peaks are our candidate clip boundaries. The vertical dashed lines in Figure 1 indicate those boundaries. If a clip between the candidate boundaries does not meet the minimum length requirement (e.g., three seconds), we keep merging it with its neighbor until the minimum length is reached. The finer dashed lines in Figure 1 (e.g., at 35.2 seconds) represent candidate boundaries that are too close to their left neighbors so that they are not used as clip boundaries. Such a segmentation leads to clips with a region below the unsuitability threshold in the middle section and regions above the threshold on both sides (see Figure 2).

Depending on the requested total length of the final video and the minimum clip length requirement, a smaller or larger portion of the middle region can be selected for inclusion in the video. For a given length of a clip to be selected, the clip with the minimal area under the unsuitability curve is chosen so that the total unsuitability of that clip is minimized. We use a default length of five seconds for video clips but we reduce that default length gradually to three

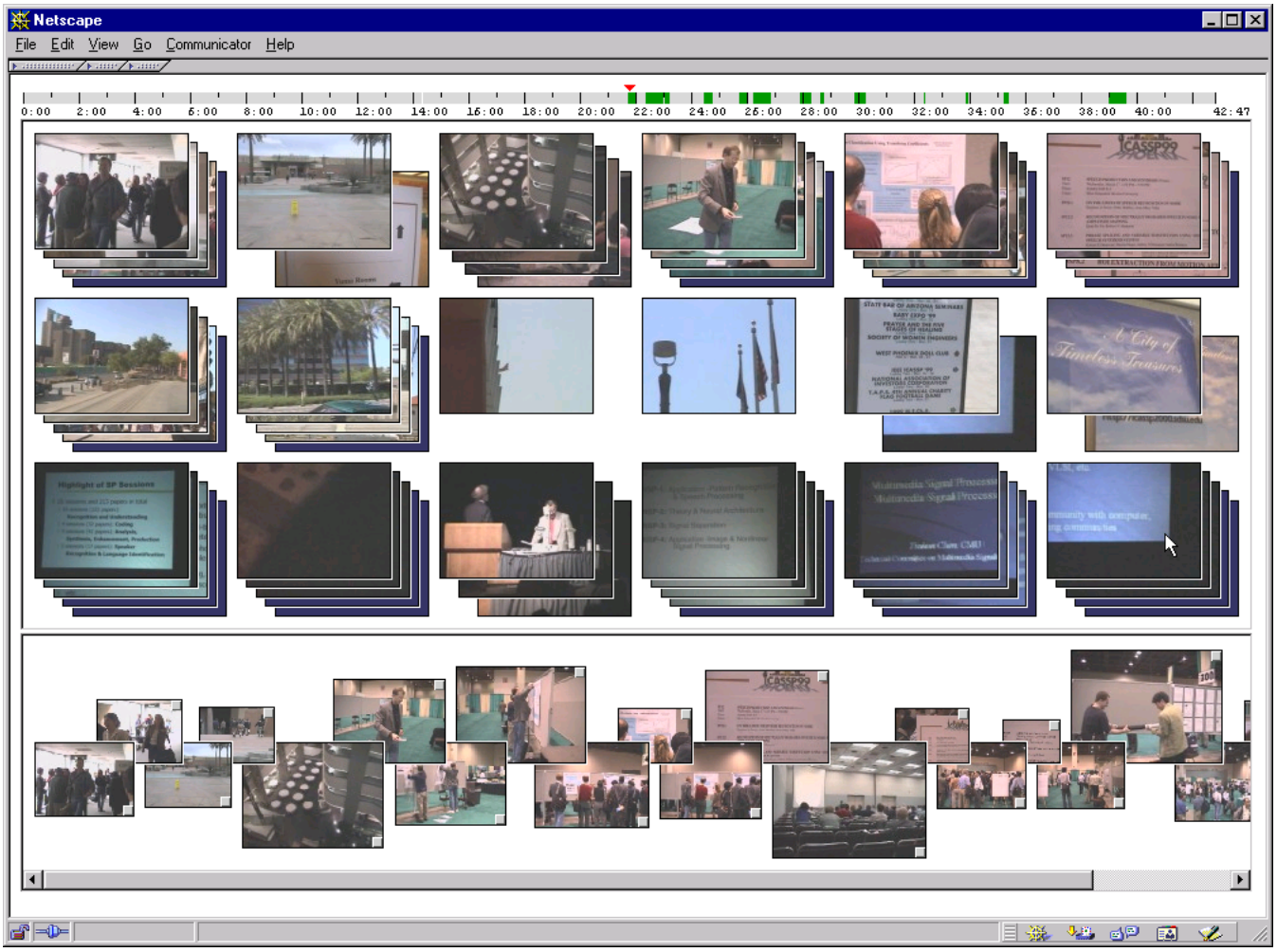

Figure 3: Video Editing User Interface

seconds for material of low quality. Requests for a larger portion can push the selected region up into the “hills” of the clip (see Figure 2). Instead of always using the clip with the minimal area under the curve, we can apply the edit rule that a clip should end with a still by pushing more into the left hills because that reduces the unsuitability (and therefore the camera motion) at the end.

Using the approach described here, most of the suitable material of a take will be included in the selected clips while the unsuitable material is relegated to the clip margins that will not be used unless the user lengthens the clip to the maximum length. A clip will contain mostly unsuitable material only if the take consists of unsuitable material. We currently consider marking such clips in the user interface or suppressing them entirely. We already suppress takes shorter than the minimum clip length.

In the opposite case of a take consisting entirely of suitable material, it might be necessary to perform an additional segmentation after the determination of clip boundaries based on peaks in the unsuitability score. For example, two stills separated by a very slow pan might show very different camera angles but would be kept together by a purely camera-motion-based segmentation. For such cases, a color-histogram-based segmentation could further subdivide a take.

Early results of a recently concluded user study indicate that users care more about the audio track than we expected. In particular, they disliked clips starting and ending in mid-sentence. We are currently integrating a silence detection algorithm that will influence clip segmentation and trimming so that clips start and end at sentence boundaries if possible.

USER INTERFACE

Once clips have been determined, they need to be made available to the user for inspection and inclusion in the output window. The user interface for Hitchcock allows the user to select the parts of the raw video to be included in the final edit and to specify the order of the material. The interface consists of two image-based displays (see Figure 3). The top display lets users select clips from the raw video. The bottom display lets the users organize the clips along the timeline and change the lengths of the clips.

We currently support the use of clips from several videos by specifying a list of video files instead of just a single one. The collection of videos is treated as a single video where

Figure 4: Flipping Through Images in a Pile

the video boundaries are considered to be camera changes. In this setup, clips from different videos might be grouped together in our clip selection interface described below. As an alternative, we consider opening videos in sequence so that the clip selection display only shows clips from a single video while the output display contains clips from several videos.

Presenting Clips

When we first envisioned our video editing system, we planned to present all identified clips to let the users choose among them. We did not consider, however, how many clips we would have to show. The sample video (about 43 minutes) shown in the top display in Figure 3 contains 225 clips. If we wanted to show all clips in the same area with the same size keyframes, it would require more than ten pages in a scrollable window. After we tried to use such an interface for a little while, we realized that it was too difficult to find the desired clips. We also noticed that several of the clips looked similar to each other and decided to use that fact.

We cluster all clips by the similarity of their color histograms and place similar clips into the same pile. Each clip is represented by one keyframe in a pile and the clips are stacked in temporal order. The FotoFile system [14] uses a similar clustering algorithm to present visually similar photographs in a Hyperbolic Tree but that approach does not appear to scale up to a large number of images. The piles are shown row-by-row in the time order of the first clip in each pile. We provide a user-selectable option to only group neighboring clips so that the display is completely in time order. In this display, however, less-similar clips may be grouped together. The Intel home video abstracting system [15] clusters clips by recording time so that clips recorded on different days are more distant than those recorded during the same hour. We plan adding that and other distance measures as user-selectable options.

The number of piles is determined by the available window area. Resizing the window changes the number of piles. Because we use a hierarchical clustering method that creates a binary cluster tree, we can select exactly as many clusters as we have piles by cutting through the cluster tree at the corresponding level. As expected, the use of a larger window with room for more piles leads to more intuitive results where truly similar images are grouped together. For our sample video, we had very good results with 30 piles.

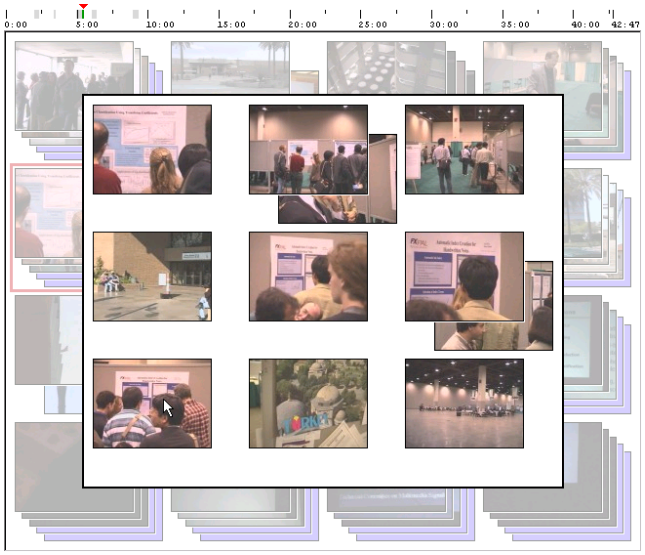

Figure 5: Expanding a Pile of Video Clips

When the user points at a pile with the mouse, the timeline above the piles shows the coverage of the clips in the pile (dark area) as well as the start time of the clip (triangle) under the mouse (see Figure 4). Furthermore, the image under the mouse is moved to the top to reveal its content. Other images in the pile are partially visible at the bottom-right corner. This type of display is similar to the one used in STREAMS [6], where the edges of video frames are shown over time. Because the images used in our display are not adjacent in time, we show more than just a one-pixel edge. If there are only a few images in the pile, we use all the available space rather than visualizing the height of the pile in perspective (the right pile in Figure 4). We also do not show more than five images in a pile. For large piles, we only show the top images and add one or two dark rectangles to indicate that there are a few more or many more images, respectively (the left pile in Figure 4).

To see the additional images in a pile, the user can expand the pile by clicking on it. The current display is faded out and the images of the pile are shown in an area in the middle of the faded out display (see Figure 5). By still showing the main display as a faded out background, the user is provided with a context for the drill-down browsing. Consequently, the user is less likely to get lost in exploring a pile. The expanded pile is marked by a border that can be seen in the faded out display (left margin in Figure 5). The timeline displays the coverage of the expanded view in light gray and the coverage of the pile in a darker color as before. This interaction style for exploring additional detail is similar to the one used in our manga video summarization approach [4]. The difference is that we need to provide access to every clip in the video for our editing system whereas the manga summary only attempts to present an overview of the video and access points to some interesting portions.

As shown in Figure 5, the display of the expanded pile might contain piles itself. Those piles can be expanded in turn (see Figure 6). The previously expanded display is now faded out; the main display is faded out even more. Taskiran

Figure 6: Repeatedly Expanding Piles

et al. [19] describe a similar cluster-based hierarchical browsing approach that supports searches in a video database instead of providing access to all the clips in a video, but the interface they describe makes it easy to become disoriented in the matrix of keyframes because it does not show the context from which drill-down occurred, and does not show the relative amount of information represented by each keyframe until that keyframe is selected.

Expanded displays can be collapsed one or more levels by clicking on the corresponding faded out area. We believe that this interaction technique is more intuitive and efficient than, for example, the use of a close box. After an expanded pile is collapsed again, it is still marked by a border so that users do not lose their place in the display. The interaction techniques described here are based on lessons learned from a user interaction study of our manga summary [4].

To determine whether a clip is really suitable for the output video, users can click on the corresponding keyframe to play that video clip (see Figure 7). Only the clip is played

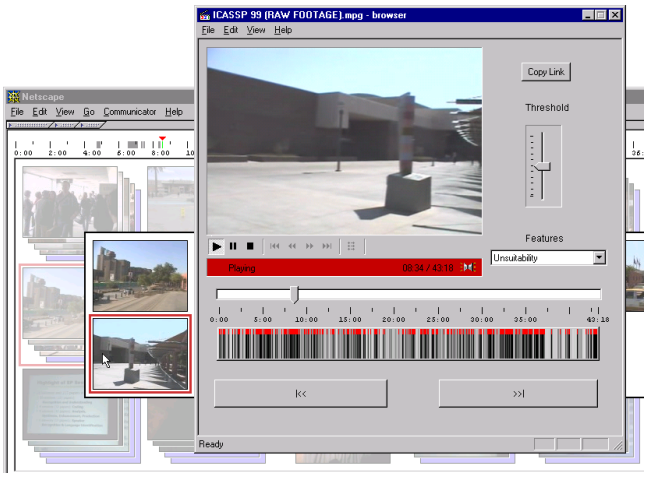

Figure 7: Playing a Video Clip

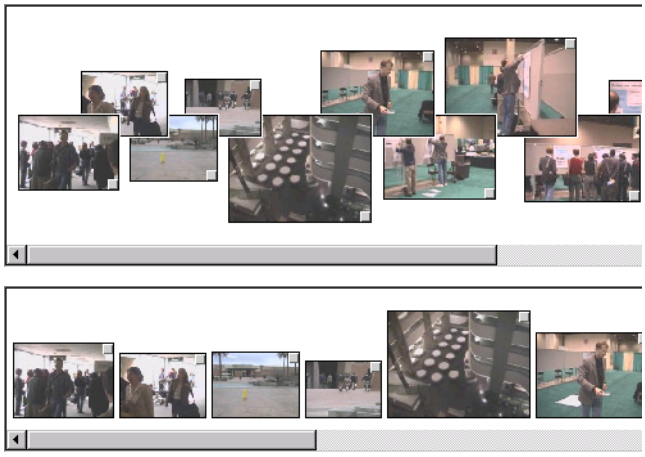

Figure 8: Two Storyboard Designs

but the user can clear the in and out points to explore the area around the clip. In our current implementation, we use the infrastructure of our MBase system [8] that lets us play MPEG and RealVideo files. Our player presents the unsuitability score as a grayscale timeline (that looks like a bar code in Figure 7). We are considering adding a zoom option for that display so that one can better judge the suitability of a short video clip.

Storyboard Timeline

The second interface display is a storyboard timeline for composing the output video. The user drags keyframes from the selection display and places them along the timeline. To indicate that a clip has been used, it is framed by a purple border in the selection window (similar to a visited link in a Web browser). The keyframes can be reordered by dragging them to different points in the timeline. Figure 8 shows two designs for that timeline. The first design saves horizontal space by staggering the images in two rows. Because of the saved horizontal space, less scrolling is required to see the whole timeline. The images are slightly overlapped to make the time order more obvious. The second design places all images in a single row. Our pilot users had strong preferences for one or the other design so that we provide both designs as a user-selectable option.

In both designs, the size of an image corresponds to the length of the clip. A handle in the corner of each image allows the user to resize the image and thereby to modify the length of the clip. Minimum and maximum length constraints are maintained by limiting the sizes of the keyframes. Early results of a recently concluded user study indicate that hard lower and upper limits of three and ten seconds, respectively, are not well received. In response, we set a much smaller lower limit (0.7 seconds) and no upper limit. The color of the resize frame turns red for clip lengths outside the recommended range of three to ten seconds. The mapping between image sizes and clip length becomes logarithmic for lengths longer than ten seconds to avoid extremely large images.

Once the representative keyframes have been selected and placed in the correct order and size in the storyboard interface, Hitchcock automatically determines the appropriate in

and out points for each clip to be included in the final customized video. The completed video can be played immediately by passing an edit decision list to the MBase media player. Individual clips can be played as well by clicking on the corresponding image. Once the user is satisfied with the generated output video, the system can create it as a single video file.

Supported Video Formats

Currently, we use the infrastructure of our MBase system [8] to play MPEG and RealVideo files. The RealVideo playback could be used in cases where a user submits a raw video to a Web-based service bureau and then edits the video while looking at a streaming video proxy. The final video could be downloaded by the user at the end. On the other hand, streaming video is not as well suited for exploring video clips because of the buffer time needed after each skip. Video clips concatenated on-the-fly would not play well with a streaming video approach. Here the user would have to wait for the single video file to be created.

One scenario that we envision is Hitchcock for home use with a DV camera connected to a PC. We expect DV to become the common format for home video. Therefore we plan to extend our infrastructure to support DV. We want our users to be able to press one button to start the DV capture and video analysis. Unfortunately, most current DV cameras only support real-time capture. Nevertheless, we expect our system to be ready for editing immediately after the capture is completed.

BALANCING CLIP LENGTHS

One significant problem users encounter when trying to compose a video of a given length is how to trade off the lengths of the various clips that make up the final result. Traditional video editing software may let the user adjust the in and out points of each clip, but then the user must examine many clips to determine ones to contract or to expand to produce the desired length.

This situation may be characterized by appealing to the theory of Cognitive Dimensions [9]. Cognitive Dimensions is a theory that describes interfaces and interactions on an abstract level; it consists of about a dozen dimensions. The clip editing operations described above may be characterized as scoring poorly on the dimensions of "hard mental operations," "role expressiveness," "viscosity," and "visibility." We examine each in turn below.

- *Hard mental operations*. The user must keep track of clips that may be shortened or lengthened outside the program (on paper, for example). This may be time-consuming and error prone.
- *Role expressiveness*. The user may not be able to determine easily the relationship between a given clip (as represented by a keyframe) and the overall video.
- *Viscosity.* A change in the length of one clip may require the user to adjust the lengths of many other clips to keep the final video within desired length constraints. Each such change then requires many additional interactions to implement and potentially to undo it.

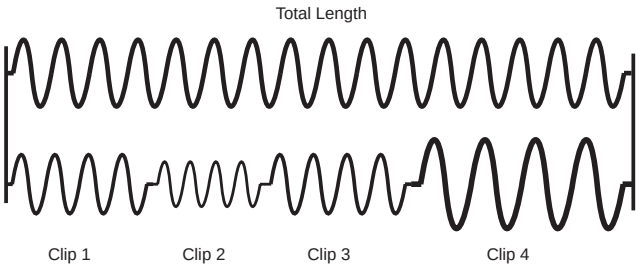

Figure 9: Springs for Clip and Total Length

- *Visibility.* The system does not provide any feedback whether lengthening a particular clip to increase the overall video length will include fairly unsuitable material while there are other clips with perfectly suitable unused portions.

Our analysis and experience with this situation suggested that a semi-automatic way of balancing clip durations to produce a video of desired length could improve the usability of the interface. While providing computer support for this activity, we wanted the user to retain creative control. To that end, we use the unsuitability score as the input for a spring model in which each selected clip has an associated spring that tries to keep it at an optimal length while dealing with the global constraint of the desired total video length (see Figure 9). The total video length is adjusted by the strength of the global spring. Individual clips can be shortened or lengthened by modifying the strengths of their associated springs. The system automatically selects the in and out points for a clip based on the strength of the associated spring.

Spring and force models have been used in several other systems. The NeXT Interface Builder [20] allows users to place springs around widgets that define how readily the widget deforms when pressed or pulled by the window it occupies. The FormsVBT dialog builder [3] employs a T_EX-based boxes-and-glue model for widget layout. The T_EX [13] typesetting system uses a numerical badness score to represent the amount a line of text is stretched or shrunk from the ideal. The system attempts to minimize the badness by changing the size of the glue joining the text. The spring model has been used for two-dimensional layout of objects (e.g., [7]). We apply similar techniques to the simpler one-dimensional (timeline) case.

Determining the Function of Spring Strength

Each clip has an associated spring that expands and contracts based on the force provided by the spring controlling the total length of the produced video (see Figure 9). Each spring of a clip has the same force applied to it so that the clip length can be determined quickly. As discussed earlier, for a given clip length the portion of the clip is selected that has the minimal area under the unsuitability curve.

The force of the spring is related to the area under the unsuitability curve. The force required to expand or to contract the spring is determined from a neutral point at which the spring is at rest. There are several possible approaches for selecting that neutral point. One option is to set the neutral length to be either zero or the minimal length of a clip

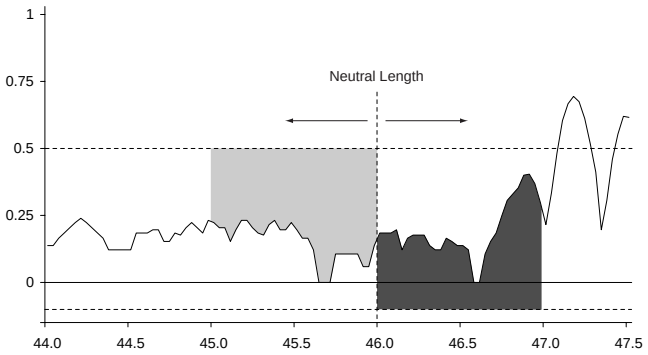

Figure 10: Determining the Spring Force

(e.g., three seconds). In this case, springs always expand and never contract. If the neutral length is equal to the minimal length, that length constraint is enforced automatically. Alternatively, the neutral length can be set to the maximum length of the clip, i.e., everything between the clip boundaries. In that case, springs always contract and never expand. As a third option, the neutral length can be the "standard" clip length (e.g., five seconds). In that case, a video without any constraints on the total length would consist entirely of five-second clips. That approach can be modified for clips that do not have a suitable portion of that length in which the unsuitability score stays below a threshold (the area under the curve does not exceed a certain value).

In selecting good clips, areas with low unsuitability scores should be kept whereas areas with high unsuitability scores should be given up easily. The force of a spring needs to be consistent with that requirement. That means that for an expanding spring, a small additional area under the curve (low unsuitability) should generate a small force (easy to pull out the spring) to get into that portion easily, whereas a large additional area under the curve (high unsuitability) should generate a large force. For a contracting spring, the opposite is true. A small area under the curve should generate a large force (difficult to push the spring in) whereas a large area should generate a small force. In the former case, the force is proportional to the area below the curve, whereas in the latter case, the force is proportional to the area above the curve (see Figure 10). In both cases, changing the length of the spring is supposed to change its force even if the area under/above the curve does not change (as can happen with an unsuitability score of zero or one). To address this concern, a constant is added to the unsuitability score that provides a linear component for the spring force function (see the area below zero in Figure 10).

For a given force, the length of the spring is determined via table lookup. During the setup phase, the system loops through all possible lengths for each clip and determines the in and out points for the clip with the smallest area under the curve. The complexity of this algorithm is $O(n^2)$ where *n* is the length of the clip in units (e.g., frames). For specifying a fixed total length (instead of modifying the strength of the global spring), the system can iteratively decrease or increase the force until the proper length is found. It might not be possible to satisfy such a request because of the minimum or maximum clip lengths. In such a case, decreasing or increasing the force will not change the total length past a certain point.

Adjusting Spring Strengths

Users can modify the strength of the spring associated with the total length of the produced video so that all individual clips are shortened or lengthened accordingly. If users want to emphasize or de-emphasize individual clips, they can manipulate the strengths of their associated springs. The effect of changing the spring strength for a clip depends on the unsuitability score curve of that clip. For a constant unsuitability score over the whole length of the clip, doubling the strength of the spring will double the length of the clip.

Integrating the Spring Model

Results of a survey conducted as part of a recently concluded user study seem to indicate that balancing clip lengths might not be that important to users. Having a video of a given total length does not seem to be a common need. We conclude from this observation that a spring model implementation should stay in the background rather than being shown explicitly. A text entry field for the total video length should be the only visible control. Using that total length, the system can iteratively modify the tension in the collection of springs until the requested length is reached. Rather than letting users modify the strengths of springs associated with individual clips, the clip length users select by resizing the image associated with the clip can be converted to the corresponding spring strength that would produce a clip of that length. With this approach, the spring model remains invisible to the user but still provides the means to balance clip lengths for a requested total video length while still allowing for adjustments of individual clips.

Summary

The spring model is a promising approach for supporting users in modifying the total output video length without having to perform many adjustments of individual clip lengths. The spring model has been added to our system but it has not been used in a recently concluded user study.

CONCLUSIONS

In this paper, we have described a novel approach for the support of home video editing. We use automatic video analysis to identify suitable clips in the raw video material. We provide a user interface that lets users explore and select those clips easily. Rather than having to determine in and out points for the clips, users can just drag keyframes representing the clips into a storyboard timeline and change the order and lengths of the clips. For a given length of a clip, Hitchcock automatically determines the appropriate in and out points.

We also proposed an approach for helping users balance clip length of the edited video without having to adjust the lengths of many individual clips. We represent the length of each clip as a spring that contracts and expands as needed to accommodate change requests for the length of the output video. The force function of each spring is determined by

the underlying suitability of portions of a video clip so that more suitable video material is preferred for inclusion in the output video.

The system described in this paper will overcome some important obstacles people currently have with editing their home video. This is important as more and more people own DV cameras and want to use PCs to create interesting video presentations from their raw video material. Early results of a user study show that users find it easy to interact with the system and to edit their own home video. At the same time, the study uncovered several conceptual and usability issues that we are currently addressing. For example, audio will play a significant role in determining clip boundaries. We also work on new modes of navigation among clips.

REFERENCES

1. Adobe. “Premiere.” http://www.adobe.com/products/premiere/
2. Apple. “iMovie.” http://www.apple.com/imovie/
3. Avrahami, G., Brooks, K.P., and Brown, M.H. “A Two-View Approach to Constructing User Interfaces.” ACM SIGGRAPH ’89, pp. 137-146, 1989.
4. Boreczky, J., Girgensohn, A., Golovchinsky, G., and Uchihashi, S. “An Interactive Comic Book Presentation for Exploring Video,” in Proceedings of CHI 2000, ACM Press, pp. 185-192, 2000.
5. Christel, M., Smith, M., Taylor, C. and Winkler, D., “Evolving Video Skims into Useful Multimedia Abstractions,” in Human Factors in Computing Systems, CHI 98 Conference Proceedings (Los Angeles, CA), New York: ACM, pp. 171-178, 1998.
6. Cruz, G. and Hill, R. “Capturing and Playing Multimedia Events with STREAMS,” in ACM Multimedia 94 Proceedings, ACM Press, pp. 193-200, 1994.
7. Fruchterman, T. and Reingold, E. “Graph Drawing by Force-Directed Placement,” in Software Practice and Experience, 21(11), 1129-1164, 1991.
8. Girgensohn, A., Boreczky, J., Wilcox, L., and Foote, J. “Facilitating Video Access by Visualizing Automatic Analysis,” in Human-Computer Interaction INTERACT ’99, IOS Press, pp. 205-212, 1999.
9. Green, T.R.G., and Petre, M. “Usability Analysis of Visual Programming Environments.” Journal of Visual Languages and Computing, 7, 131-174, 1996.
10. He, L., Sanocki, E., Gupta, A., and Grudin, J. “Auto-Summarization of Audio-Video Presentations”, ACM Multimedia ’99, pp. 489-498, 1999
11. In-Sync. “Speed Razor.” http://www.in-sync.com/
12. Javu Technologies. “JavuNetwork.” http://www.javu.com/
13. Knuth, D., *The $T_EX$book*, Addison Wesley, 1984.
14. Kuchinsky, A., Pering, C., Creech, M.L., Freeze, D., Serra, B., and Gwizdka, J. “FotoFile: A Consumer Multimedia Organization and Retrieval System,” in CHI 99 Conference Proceedings, ACM Press, pp. 496-503, 1999.
15. Lienhart, R. “Abstracting Home Video Automatically,” ACM Multimedia 99 Proceedings (Part 2), pp. 37-40, 1999.
16. Pfeiffer, S., Lienhart, R., Fischer, S. and Effelsberg, W., “Abstracting digital movies automatically,” in Journal of Visual Communication and Image Representation, 7(4), pp. 345-353, December 1996.
17. Russell, D. “A Design Pattern-Based Video Summarization Technique Moving from Low-Level Signals to High-Level Structures.” In Proceedings of the Thirty-Third Annual Hawaii International Conference on System Sciences (HICSS-33), pp. 84, 2000.
18. Smith, M. and Kanade, T., “Video Skimming and Characterization through the Combination of Image and Language Understanding Techniques,” in Proceedings of Computer Vision and Pattern Recognition, pp. 775-781, 1997.
19. Taskiran, C., Chen, J.-Y., Bouman, C.A., and Delp, E.J. “A Compressed Video Database Structured for Active Browsing and Search,” in ICIP’98, vol. 3, pp 133-137, 1998.
20. Webster, B.F. *The NeXT Book*. Addison-Wesley, 1989.