Video Editing Using Lenses and Semantic Zooming

A. Chris Long, Brad A. Myers, Juan Casares, Scott M. Stevens, and Albert Corbett
Human Computer Interaction Institute,
Carnegie Mellon University
Pittsburgh, PA 15213,USA
Tel: 1-412-268-7565
E-mail: chrislong@acm.org

ABSTRACT
Digital video is becoming increasingly prevalent. Unfortunately, editing video remains difficult for several reasons: it is a time-based medium, it has dual tracks of audio and video, and current tools force users to work at the smallest level of detail. Based on interviews with professional video editors and observations of the use of our own editor, we have developed new interaction techniques for digital video based on semantic zooming and lenses. When copying or cutting a piece of video, it is desirable to select both ends precisely. However, although many video editing tools allow zooming into a fine level of detail, they do not allow zooming at more than one location simultaneously. Our system provides multiple lenses on the same timeline, so the user can see more than one location in detail.

KEYWORDS: Digital video editing, multimedia authoring, metadata, Silver, lenses, focus+context, zooming

INTRODUCTION
Digital video is becoming increasingly ubiquitous. Digital video equipment is more accessible than ever, and there is an increasing amount of video material available on the World Wide Web and in digital libraries. As technology and content become available, users will shift from passive spectators to active creators of video. Forrester Research predicts that by 2005, 92% of online consumers will create personal multimedia content at least once a month [24]. However, although many exciting research projects are investigating how to search, visualize, and summarize digital video, there is little work on new ways to support the use of the video beyond just playing it.

This is unfortunate, because video editing has several unique challenges not found with other media. One is that digital video is a time-based medium. This property makes it difficult for users to browse and skim video. Users often must linearly search their source video to find the clip they desire.

Another challenge for editing video is that it is a dual medium. Most "video" actually consists not just of a video track but also an audio track. These tracks must be kept synchronized, but the user must also be able to edit them separately when desired, for example during transitions from one shot to another. Further, when a shot is cut from a video for use elsewhere, the user must be able to disentangle overlaid audio and video.

A third problem is that the syntactic units that users want to edit are usually shots of video and words or sentences of audio, but current tools require users to examine video at the individual frame level and audio using a waveform. To perform most editing operations, such as cutting a shot, the user must manually pinpoint specific frames, which may involve zooming and numerous repetitions of fast-forward and rewind operations. Finding a specific word or sentence using a waveform is similarly tedious.

These problems make editing video a difficult, tedious, and error-prone activity. Commercially available tools, such as Adobe Premiere [1] and Apple iMovie [2], allow creation of high-quality video, but they do not adequately address the issues raised above, which makes them hard to use, especially for novices.

To better understand video editing, we visited a video processing studio and interviewed professional video editors. We also examined commercial and research video editing systems. Finally, we ran a study of subjects using our own preliminary video editor and noticed what tasks were most difficult [8].

To help solve the observed problems, we are developing a new video editor as part of the Silver project. Silver stands for **S**implifying **I**nterface **L**ayout and **V**ideo **E**diting and **R**euse. The goal of Silver is to make editing of video as

Submitted for Publication

easy and widespread as it is today to edit formatted text using a modern word processor. A previous version of the Silver editor was presented earlier [8]. The current paper describes an entirely new implementation: Silver2. The innovations for video editing in Silver2 include a timeline view that visualizes the video at multiple resolutions with *semantic zooming*, and has the ability to provide multiple "lenses" [6] that overlay the timeline view and allow the user to work at multiple resolutions at the same time. The lenses will enable users to more easily zoom in to view the details of the two ends of a segment of video to be cut. Silver2 can automatically place these lenses when the user selects a region of video that will not all fit on the screen at a frame-level resolution.

The remainder of the paper is organized as follows. First, we discuss related work. Next, we describe the Silver2 interface. We end with conclusions and future work.

RELATED WORK

In this section, we describe different types of related work. We start with systems that use metadata for video editing. Next, we discuss systems that automate the process of video editing to some degree. Then we discuss work on video visualizations that address the issues of scale and time. Next, we describe previous work on lenses, followed by zooming interfaces. We conclude with a description of the previous version of Silver.

Metadata and Video Editing

The notion of using metadata for editing video is not new. For example, Mackay and Davenport examined the role of digital video in several interactive multimedia applications and concluded that video is an information stream that can be tagged, edited, analyzed, and annotated [20]. Later, Davenport et al. proposed using metadata for home movie editing assistance [11]. However, they assumed this data would be obtained through manual logging or with a "data camera" during filming, whereas our system creates the metadata automatically using the Informedia digital video library [17].

Currently, there is a large body of work on the extraction and visualization of information from digital video (e.g., [14, 26]) that make a data camera unnecessary. One example of a system that uses metadata is IMPACT [28], which uses automatic cut detection and camera motion classification to create a high level description of the structure of the video. The user can organize the shots in a tree structure and then edit the composition by moving the branches [27]. IMPACT supports this process by recognizing objects across multiple shots.

Automation

Several systems edit video with varying degrees of automation. Fully automated systems may be used for news on demand [3] or quick skimming [8], but do not really support authoring.

The Video Retrieval and Sequencing System (VRSS) [10] semiautomatically detects and annotates shots for later retrieval. Then, a cinematic rule-based editing tool sequences the retrieved shots for presentation within a specified time constraint. Examples of cinematic rules include the parallel rule, which alternates two different sets of shots, and the rhythm rule, which selects longer shots for a slow rhythm and shorter shots for a fast one.

The Hitchcock system [15] automatically determines the "suitability" of the different segments in raw home video, based on camera motion, brightness, and duration. Similar clips are grouped into "piles." To create a custom video, the user drags segments into a storyboard and specifies a total desired duration and Hitchcock automatically selects the start and end points of each clip based on shot quality and total duration. Clips in the storyboard are represented with frames that can be arranged in different layouts, such as a "comic book" style layout [7].

The MovieShaker system automates video editing almost completely [25]. The user specifies multiple video clips and a "mood" (e.g., romantic, wild), and the program combines and edits the clips into a single video.

While automation makes editing faster, it usually involves taking away power from the user, which is not always desirable. In fact, user studies led to changes in Hitchcock to give more information and control to the user [16]. Our system does not try to automate the video processing, instead concentrating on easing the authoring process for people.

Visualizing Time and Scale

Video editing is difficult in part because it is hard to visualize time. Due to the semantic and syntactic nature of the video, where selecting a five-second piece involves fiddling with frames in $1/30^{th}$ of a second increments, it becomes important to present the information at different scales. A single scale is not sufficient to efficiently perform all the operations needed. For example, when looking at the individual frames at a reasonable size, at most about 2 seconds of video will fit on the screen (60 frames), yet most video segments are likely to be many minutes or hours long (e.g., 18,000 to 162,000 frames). The individual frame level view is needed to make edits precisely, but more global views are necessary to find appropriate segments and to understand the context.

The Hierarchical Video Magnifier [21] allows users to work with a video source at fine levels of detail while maintaining an awareness of the context. It provides a timeline to represent the total duration of the video source; a second timeline shows a detail of the first one and

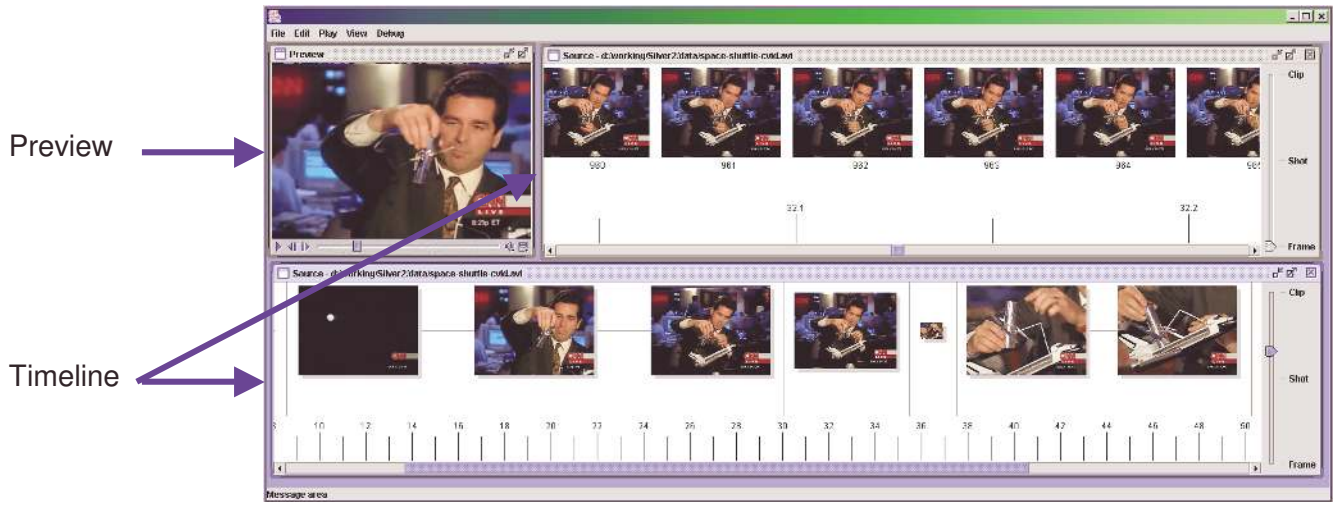

Figure 1 Silver2 overview

illustrates it with a frame sampling. The user can select which portion of the top timeline that is expanded on the next one. There is also a mechanism that can be used to add a new timeline with a higher level of detail. Successive timelines create an explicit spatial hierarchical structure of the video source.

The Swim Hierarchical Browser [0] significantly improves on this idea by using metadata. Swim displays automatically detected shots in the higher level layers instead of frame samples.

Lenses

The idea of a "magic lens" was introduced by Eric Bier, *et. al.* [6] in 1993, and there have been many subsequent systems that have built on this concept (e.g., [12][18][23]). A magic lens is a screen region that transforms the objects underneath it. Typically, the user can move the lens to control what object or screen region is affected, and can often change properties of the lens. Lenses have been used to provide various transformations and visualizations of the content underneath and to provide alternative ways of interacting with the underlying content [4]. Rather than just zooming the content like a magnifying glass, many lenses provide semantic transformations of the content to show and allow manipulation of the underlying properties. For example, Hudson's debugging lenses [18] can show the class name and extent of user interface widgets, children widgets, etc. Our visualization is closest to the document lens [23] because it shows the main area of interest larger, and rather than occluding the neighboring areas, it shows the neighboring areas smaller. This is a form of "fish-eye view" [13], also called a "focus+context" display [23]. As far as we know, Silver2 is the first application of lenses to video editing.

Zooming

The Pad, Pad++, and Jazz projects [4][5] have been researching zoomable interfaces, where the user can zoom in and out of the content to see more detail or more context. Often, zooming will not just change the magnification, but may change the form and style of the content so it is appropriate for the current zoom level. This is called "semantic zooming" and has been used by a number of other systems (e.g., [19][0]). Our approach is closest to DENIM [19], in which a slider provides continuous zooming that jumps to a different representation at various points on the slider.

Silver Version 1

The current Silver2 system is based on experience with building a previous version [8][22]. The earlier version, implemented in Visual Basic, provided multiple views of the video, including a storyboard, project, timeline, transcript, hierarchy, and preview views. In the timeline, interaction techniques supported sophisticated zooming and panning. The system could automatically adjust selections and regions that were cut and pasted based on the underlying content of the video and audio. In user tests, however, we found a number of problems with the system, which the new Silver2 interface seeks to address. In particular, users still found it tedious and difficult to get both ends of a selection to be in exactly the right places for edits.

SILVER2 INTERFACE

An overview of the timeline and preview views of the Silver2 interface is shown in Figure 1. There can be multiple simultaneous views of the same video composition. Each view can show a different part of the video at a different zoom level. In this example, the top timeline shows individual frames of video. The bottom timeline has been zoomed out to show the *shot* level of

Figure 2 The zoom slider.

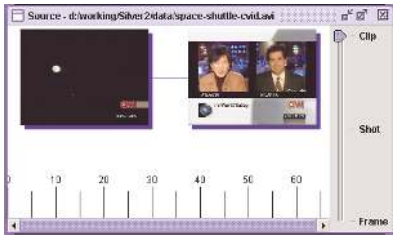

Figure 4 Clip zoom level

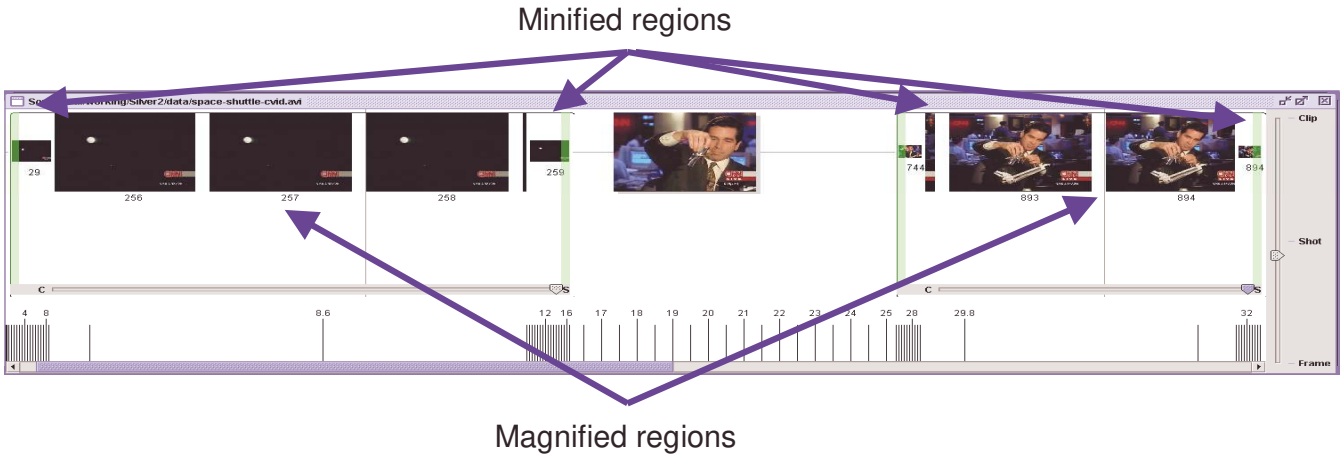

Figure 3: Two lenses at the beginning and ends of a shot

detail. A *shot* is a continuous segment of video from a single camera.

Silver2 is implemented in Java using the Swing and Java Media Framework toolkits. Silver2 uses the Informedia digital library [8][26][29] as the source for the video clips, and to provide the metadata about the video content. This metadata includes where shot breaks are and the transcript for the audio.

The remainder of this section describes details of the Silver2 interface design. The first subsection describes zooming, the second describes lenses, and the third describes smart selection.

Zooming

Like other video editors, Silver2 allows the user to view video at multiple temporal resolutions and at multiple levels of detail. Unlike previous video systems, however, the user can easily change both of these properties by continuously sliding a single slider (see Figure 2). This combination of scaling-based zooming and semantic zooming was inspired by DENIM [19], which had a similar mechanism for navigating website designs.

When the slider is at the bottom, the timeline displays individual frames of video. As the slider is dragged up, frames are left out so every *n*th frame is shown. As the slider moves closer to the "Shot" tick mark than the "Frame" mark, the view changes to show shots of video rather than frames (see Figure 1, bottom timeline). Shots are shown in differing amounts of detail depending on the amount of screen space available. If there is enough space for only one frame, a representative thumbnail frame is shown, scaled down if it would not fit full-sized.[1] If there is space for two frames, then the first and last frames are shown. If there is enough space for three or more frames, then the first, last, and thumbnail frames are shown. Gray lines are used to connect frames within a shot and to separate shots from one another. Each frame drawn in this view has a gray drop shadow to indicate it represents a shot.

[1] A representative frame for each shot is provided by the Informedia digital video library.

When the slider is closest to "Clip," the timeline shows *clips*[2] of video in a similar manner to shots (see Figure 4). The first and last frames of a clip are shown, unless the clip is too short, in which case the first frame is shown. When the slider is all the way at the top, the entire composition is shown in the timeline. Connecting lines and drop shadows are blue to distinguish clips from shots.

We assume that the user is most interested in the region in the center of the window, so zooming keeps the view centered on the same time, and changes both ends.

Lenses

With multiple timelines at different zoom levels, the user can view video in fine detail and see an overview at the same time. However, we found in our previous system that manipulating multiple timelines to do detailed editing can be tedious, because of the additional scrolling and zooming that it entails.

Lenses are easier to use because they provide the detail in the context of the whole. For example, lenses allow the beginning and the end of a shot to be shown in detail simultaneously on the same timeline (see Figure 3).

We assume that the user places lenses at specific points on the timeline because those are the points of interest. Therefore, interactions with other parts of the interface, or other lenses, should not change a lens's focus, its duration, or its screen size. This principle informed our many decisions about how the lens should behave when interacting with the user, the surrounding timeline, and other lenses.

For example, there are several ways a lens could magnify its content and interact with the surrounding timeline. One option is to behave like a magnifying glass and show the area under its center magnified but obscure areas under its border. Another is to avoid obscuring anything by expanding the timeline outside of the lens. The effect of this is to "push" the content not being viewed in the lens out of the lens area. This option has the disadvantage that inserting, removing, or resizing a lens would affect the entire timeline. We chose a strategy inspired by fisheye lenses [13][23]: the lens magnifies a central, or *focus* region, and minifies regions at its edges to compensate. The result is that the lens as a whole occupies exactly the same screen space as the normal, lens-less view would.

More specifically, we devote a fixed fraction of the width of each lens (in pixels) to the focus, and the rest to the minified edge regions (currently, 80% for the focus and 10% for each edge). The zoom level of the focus is specified by the user. The zoom levels of the edges are computed so that they accommodate the amount of time required to fit the duration of the entire lens into its width, which depends upon the zoom level of the outside timeline. For example, if the zoom level of the lens is only slightly different from the timeline zoom level, the edges only need to be slightly minifed, but if the lens is greatly zoomed in compared to its timeline, the edges need to be greatly minified to compensate.

Most of the time, the focus of a lens is at its center. However, some manipulations (described below) can cause the focus to be off-center (e.g., see Figure 6). Since we want to keep the size and duration of the lens constant, when the lens focus shifts, each edge must change two of its three properties (i.e., width, zoom level, and duration). We decided to keep the width constant, so as the focus shifts toward one edge of the lens, that edge displays less duration in finer detail. This behavior ensures that some context information will always be visible next to the focus and furthermore shows the edge in which the focus is moving in increasing detail, which is beneficial because the user is presumably more interested in the content in that direction.

The behavior of a lens when its zoom level or the zoom level of its timeline changes is determined by our desire to maintain user decisions about lens placement and zoom level. If the user changes the zoom level of the lens, its focus changes to the level set by the user, and the duration of the focus changes to whatever fits, given its size. The position of the focus region stays centered, so like the timeline as a whole, the lens zooms from the center. The zoom and duration of the edges are changed so the lens as a whole covers the same time as a lens-less timeline would. When the zoom level of the timeline changes, the lens focus remains constant and its edges are updated.

Another design decision was how a lens should be dragged and resized. The naïve method would be to drag it in the coordinates of the timeline in which it is embedded. However, typically the lens is zoomed in to a much finer granularity than its surrounding timeline, so moving even a pixel or two would usually move the lens focus a great distance. By default, we chose resize and drag to operate at the scale of the lens focus, which allows the user to easily make fine adjustments in the lens position or size. This strategy has the disadvantage that the resize or move control that the mouse started on will not stay under the mouse as it is dragged, but we believe this tradeoff is acceptable. If the user wishes large-scale control, a modifier key can be used to manipulate the lens at the timeline scale, in which case the control and mouse stay together on screen. Lens controls are shown in Figure 5.

Since a vertical slider would be too visually disruptive on the timeline, a horizontal one is provided to allow the user to adjust the zoom level of the lens. In theory, one could

[2] A clip is a continuous segment of video from a single source file.

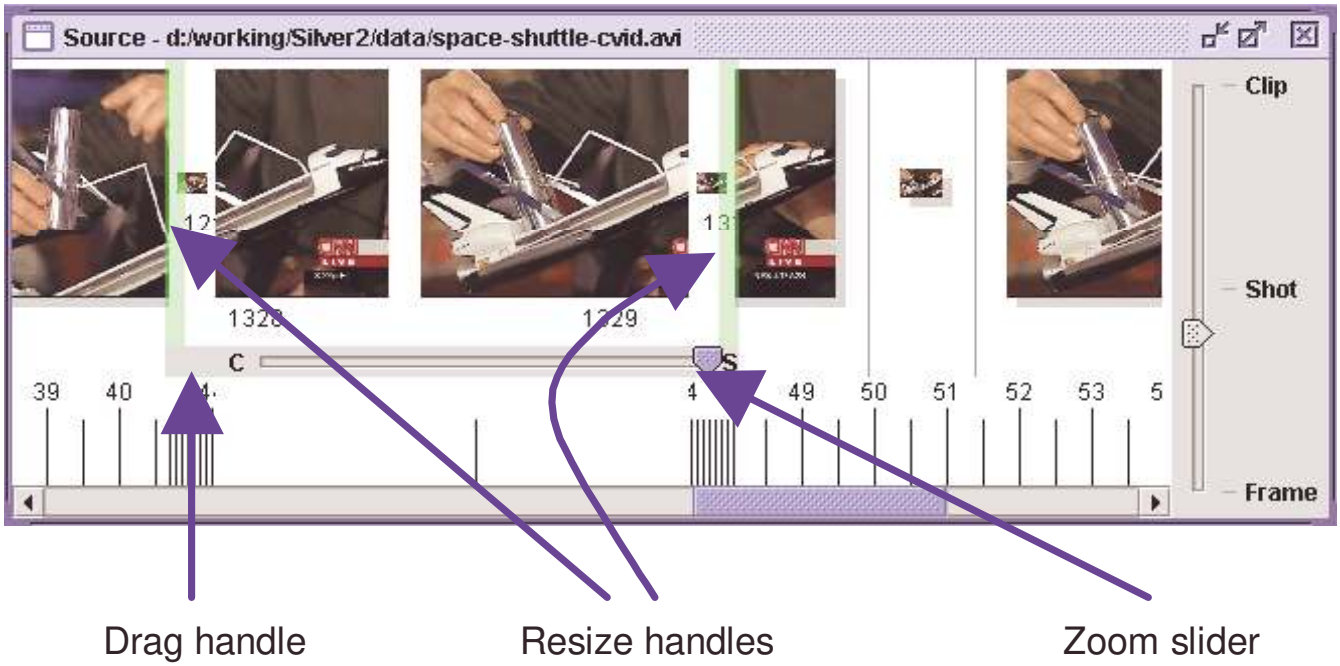

Figure 5 Lens controls

have a lens that minified its focus rather than magnifying it, but this behavior does not seem useful. If a user tries to zoom out a lens farther than its surrounding timeline, it stops zooming when the inside zoom factor matches the surrounding timeline.

Normally, the center of the lens focus is the same as the center of the whole lens. However, when the lens is near one end of the timeline, this policy is inadequate, because the region near the boundary will always be in the minified region, never in the focus. To solve this problem, when a lens is pushed against an end of the timeline, its focus is shifted away from the center of the lens toward the boundary. Once the focus has shifted far enough that the edge and focus are at the same zoom level, the edge region is not necessary and it vanishes (see Figure 6).

Interaction between lenses was another design issue. We discussed allowing overlapping lenses [12], but it does not seem useful for our application, and it would be difficult to control overlapping lenses since their controls for moving, growing, and zooming would overlap. Instead, when a lens is pushed against another lens, it pushes the lens, unless it is at the end of the timeline, in which case nothing happens. If

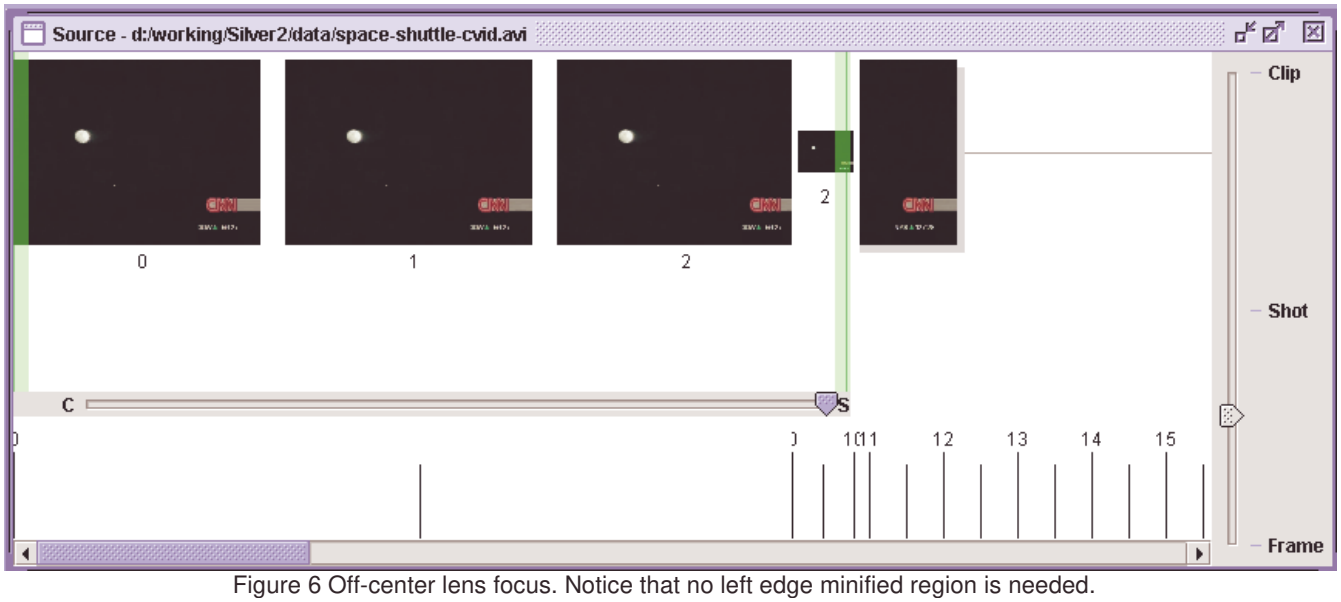

Figure 6 Off-center lens focus. Notice that no left edge minified region is needed.

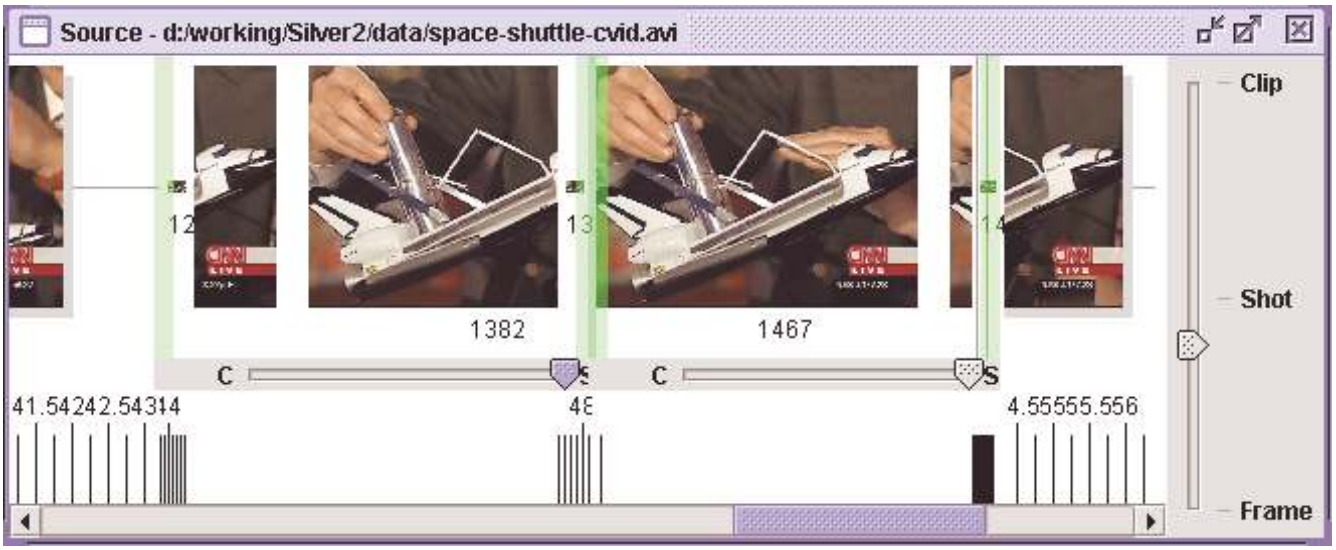

Figure 7 Adjacent lenses create a larger, zoomed, seamless view

the two lenses are at the same zoom level, dragging them together can produce a larger, zoomed, seamless view of the timeline (see Figure 7).

These interactions of lenses with the timeline ends and with each other mean that by suitable zooming of the timeline and placement of lenses, we can always see both ends of a selection in detail, no matter how near or distant they are. The underlying content flows into and out of the zoomed region in a smooth, natural way.

Another design decision is whether the lens should move with the video when the timeline scrollbar is used, or if it should "stick" to the display. Both behaviors are sometimes useful, but we decided the lens should stay with its video. This is more useful in the case when the lens is being used to focus on one end of an intended edit position, so the user can scroll and zoom to find the other end without changing the first end. Furthermore, if the user wants the other behavior and is surprised by this outcome, it is much easier to recover from this result than if the lens were focused on an interesting piece of video and scrolling the timeline moved the video out of the lens, since then the focused area is lost.

A lens can be created by selecting a region in the timeline and using a menu command. However, lenses will frequently be needed for examining shot boundaries, so when the timeline is showing shots, a modified double-click creates a lens that is centered on the nearest shot boundary and zoomed in to the individual frame level. These automatically generated lenses are initially half the size of the screen, or less if the shot boundary is near the end of the timeline.

Silver can also automatically generate lenses based on the selection. This operation creates a new timeline that is zoomed out far enough to show both ends of the current selection and at each end of the selection a lens is created and centered on the end. The lenses are set at the frame level of detail and large enough to show a few frames.

Semantic Selection

Like the original Silver system [22], Silver2 has a selection feature based on video semantics. A double-click in the timeline selects the current "unit" of video. For example, when the video is at the frame level of detail, a double-click selects a frame. When it is at the shot level, a double-click selects a shot. This interaction works through a lens as expected. That is, if the timeline is at shot level but the lens is at the frame level, double-clicking inside the lens selects a frame. As with dragging and resizing the lens, the user can use a modifier key to override the lens zoom level and use the zoom level of the timeline.

CONCLUSIONS AND FUTURE WORK

We believe these new visualizations of video will be helpful for detailed video editing. The semantic zoom unifies different semantic levels of detail with traditional magnification-based zooming, which makes it easy to change between levels of detail.

With current tools, it is difficult to view individual frames without getting lost in the sea of frames, but Silver2 allows users to view details at multiple places in the same composition, either on the same or different timelines. Lenses can be easily pushed together and separated again, as the users' needs demand.

There are many areas for future improvement in Silver2:

- To include transcripts of the audio in the timeline.
- To allow semantic zooming of the transcript, using coarse levels of detail based on sentences and speaker changes.

- To allow the text and video to be zoomed at different levels of detail simultaneously, to address the problem of differences in scale between audio and video.
- To provide intuitive selection and editing operations, such as modern word processors provide for text. For example, drag-and-drop, cut, and paste.
- To change how lenses interact. The current interaction (i.e., one lens pushing another) may not be ideal. We would like to experiment with other techniques, such as having a lens "steal" time from its neighbor when dragged next to it.
- To evaluate Silver2 with users.

We believe the framework Silver2 provides will enable editing that is easier to use than professional tools, yet more powerful than current simple consumer tools.

ACKNOWLEDGMENTS

The authors would like to thank Rishi Bhatnagar, Laura Dabbish, and Dan Yocum for their work on previous versions of Silver.

The Silver Project is funded in part by the National Science Foundation under Grant No. IIS-9817527, as part of the Digital Library Initiative-2. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the National Science Foundation.

REFERENCES

1. Adobe Systems Incorporated. Adobe Premiere. http://www.adobe.com/products/premiere/main.html.
2. Apple Computer, Inc. iMovie. http://www.apple.com/imovie/.
3. Ahanger, G. and Little, T. D. C. "Automatic Composition Techniques for Video Production." IEEE Transactions on Knowledge and Data Engineering, 10(6):967-987, 1998.
4. Bederson, B., Hollan, J.D., Perlin, K., Meyer, J., Bacon, D., and Furnas, G., "A Zoomable Graphical Interface for Exploring Alternate Interface Physics." *Journal of Visual Languages and Computing*, 1996. 7(1): pp. 3-31.
5. Bederson, B.B., Meyer, J., and Good, L. "Jazz: an extensible zoomable user interface graphics toolkit in Java," in *UIST'2000: Proceedings of the 13th annual ACM symposium on User interface software and technology*. 2000. San Diego, CA: pp. 171-180.
6. Bier, E.A., Stone, M.C., Pier, K., Buxton, W., and DeRose, T.D. "Toolglass and Magic Lenses: The See-Through Interface," in *Proceedings SIGGRAPH'93: Computer Graphics*. 1993. 25. pp. 73-80.
7. Boreczky, J., Girgensohn, A., Golovchinsky, G., and Uchihashi, S. "An Interactive Comic Book Presentation for Exploring Video." CHI Letters: Human Factors in Computing Systems (SIGCHI), 2(1):185–192, April 2000.
8. Casares, J. Myers, B, Long, A. C., Bhatnagar, R. Stevens, S., Dabbish, L., Yocum, D. and Corbett, A. "Simplifying Video Editing Using Metadata," in *Proceedings of Designing Interactive Systems (DIS 2002)*, London, UK, June 2002 (to appear).
9. Christel, M., et al., "Techniques for the Creation and Exploration of Digital Video Libraries." Chapter 8 of Multimedia Tools and Applications, B. Furht, ed. Kluwer Academic Publishers. Boston, MA. 1996.
10. Chua, T. and Ruan, L. "A video retrieval and sequencing system." ACM Transactions on Information Systems, 13(4):373–407, 1995.
11. Davenport, G., Smith, T.A., and Pincever, N. "Cinematic Primitives for Multimedia." IEEE Computer Graphics & Applications, 11(4):67–74, 1991.
12. Fox, D. "Composing magic lenses," in *Proceedings SIGCHI'98: Human Factors in Computing Systems*. 1998. Los Angeles, CA: pp. 519 - 525.
13. Furnas, G.W., "Generalized fisheye views," in *Proceedings of CHI'86 Conference on Human Factors in Computing Systems*, 1986, ACM. Boston. pp. 16-23.
14. Gauch, S., Li, W., and Gauch, J. "The VISION Digital Video Library." Information Processing & Management, 33(4):413–426, 1997.
15. Girgensohn, A., Boreczky, J., Chiu, P., Doherty, J., Foote, J., Golovchinsky, G., Uchihashi, S., and Wilcox, L. "A semi-automatic approach to home video editing." CHI Letters: Symposium on User Interface Software and Technology (UIST), 2(2):81–89, 2000.
16. Girgensohn, A., Bly, S., Shipman, F., Boreczky, J. and Wilcox, L. "Home Video Editing Made Easy—Balancing Automation and User Control." In Human-Computer Interaction INTERACT '01. IOS Press, 464–471, 2001.
17. Hauptmann, A. and Smith, M. "Text, Speech, and Vision for Video Segmentation: The Informedia Project." In AAAI Symposium on Computational Models for Integrating Language and Vision, Cambridge, MA, Nov. 10–12, 1995.
18. Hudson, S., Rodenstein, R., and Smith, I. "Debugging Lenses: A New Class of Transparent Tools for User Interface Debugging," in *Proceedings UIST'97: ACM SIGGRAPH Symposium on User Interface Software and Technology*. 1997. Banff, Alberta, Canada: pp. 179-187.
19. Lin, J., Newman, M.W., Hong, J.I., and Landay, J.A. "DENIM: finding a tighter fit between tools and practice for Web site design," in *Proceedings of CHI'2000: Human Factors in Computing Systems*. 2000. The Hague, The Netherlands: pp. 510-517.
20. Mackay, W.E. and Davenport, G., "Virtual Video Editing in Interactive Multimedia Applications." Communications of the ACM, 32(7):832–843, 1989.
21. Mills, M., Cohen, J., and Wong, Y. "A Magnifier Tool for Video Data," in SIGCHI '92 Conference Proceedings of Human Factors in Computing Systems. 1992. Monterey, CA: ACM. pp. 93–98.

22. Myers, B., Casares, J., Stevens, S., Dabbish, L., Yocum, D. and Corbett, A., “A multi-view intelligent editor for digital video libraries”, in Proceedings of the first ACM / IEEE joint conference on digital libraries, 2001, pp. 106–115.

23. Robertson, G.G. and Mackinlay, J.D. “The document lens,” in *Proceedings UIST'93: ACM SIGGRAPH Symposium on User Interface Software and Technology*. 1993. Atlanta, GA: pp. 101-108.

24. Schwartz, J., Rhinelander, T. and Dorsey, M., “Personal Rich Media Takes Off”, The Forrester Report, Forrester Research Inc., October 2000.

25. Sony Electronics, Inc. MovieShaker. http://www.ita.sel.sony.com/jump/movieshaker/ms.html.

26. Stevens, S.M., Christel, M.G., and Wactlar, H.D., “Informedia: Improving Access to Digital Video.” Interactions: New Visions of Human-Computer Interaction, 1994. 1(4): pp. 67–71.

27. Ueda, H. and Miyatake, T. “Automatic Scene Separation and Tree Structure GUI for Video Editing,” in Proceedings of ACM Multimedia '96. 1996. Boston:

28. Ueda, H., Miyatake, T., Shigeo Sumino and Akio Nagasaka. “Automatic Structure Visualization for Video Editing,” in Proceedings of INTERCHI'93: Conference on human factors in computing systems. 1993. Amsterdam: ACM. pp. 137–141.

29. Wactlar, H.D., et al., “Lessons learned from building a terabyte digital video library.”, IEEE Computer, 1999. 32(2): pp. 66–73.

Zhang, H., et al. “Video Parsing, Retrieval and Browsing: An Integrated and Content-Based Solution,” in *ACM Multimedia 95: Proceedings of the third ACM international conference on Multimedia*. 1995. San Francisco, CA: pp. 15–24.