MiniMedia Surfer: Browsing Video Segments on Small Displays

Maryam Kamvar*, Patrick Chiu, Lynn Wilcox, Sandeep Casi, Surapong Lertsithichai

FX Palo Alto Laboratory
3400 Hillview Ave., Bldg. 4
Palo Alto, CA 94304
{lastname}@fxpal.com

*Columbia University
Department of Computer Science
1214 Amsterdam Ave.
New York, NY 10027
mkamvar@cs.columbia.edu

Abstract

It is challenging to browse multimedia on mobile devices with small displays. We present *MiniMedia Surfer*, a prototype application for interactively searching a multimedia collection for video segments of interest. Transparent layers are used to support browsing subtasks: keyword query, exploration of results through keyframes, and playback of video. This layered interface smoothly blends the key tasks of the browsing process and deals with the small screen size. During exploration, the user can adjust the transparency levels of the layers using pen gestures. Details of the video segments are displayed in an expandable timeline that supports gestural interaction.

Categories & Subject Descriptors: H.5.1. [**Information Interfaces and Presentation (e.g., HCI)**]: Multimedia Information Systems – *Video (e.g., tape, disk, DVI)*

General Terms: Design, Human Factors.

Keywords: Small displays, pen-based computers, video, multimedia.

INTRODUCTION

Digital video for training and instruction is beginning to replace textual manuals. Often, workers need to access this video outside the office or workplace using a mobile device such as a PDA or cellphone. For example, a sales representative may need information about a new product to answer a customer's questions. If the information is stored in a video database, the sales representative can query the database to find the segment of video relating to the product, view it on a mobile device, and relate it to the customer.

The small displays on these mobile devices can make it difficult to access information using standard video interfaces. Displaying a list of query results with keyframes plus text or with storyboards (e.g. [4], [7], [11]) would be ineffective since the images would be too small to convey much information.

In this paper, we describe the interaction design issues that arose from prototyping *MiniMedia Surfer*, which is an application for browsing a database of video segments on mobile devices with small displays. Unlike many search interfaces that treat the process of extracting relevant information from a database as three isolated tasks (formulating a keyword query, exploring the results of the query, and viewing the desired document), MiniMedia Surfer smoothly blends these tasks by employing a layered interface. This integration of subtasks makes the interface more consistent because the selection targets and widget placements remain stable, and the transitions between tasks becomes more fluid.

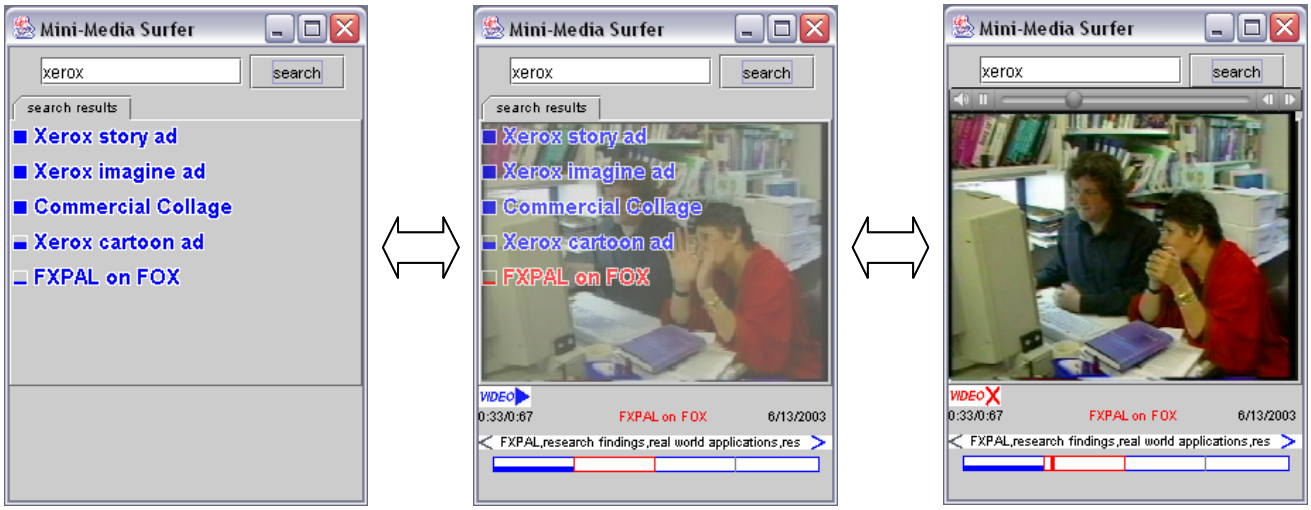
Using transparent layers, information can be organized and put more densely on the screen. Studies have shown that at appropriate levels of transparency, layers can be effectively distinguished [5]. Different layers have been used to hold separately the content and the widgets ([2], [6], [8]), or to view multiple documents of the same type ([1], [10]). In contrast, with MiniMedia Surfer the layering is task-based: there are layers for keyword query, keyframe exploration, and video playback.

VIDEO SEGMENTS DATABASE

Before going further into the user interface aspects of MiniMedia Surfer, we first need to give a brief description of the structure of the video segments in the database. The database contains original or raw video footage segmented into informational chunks (see Fig. 4). Segmentation can be performed manually or automatically via a video segmentation algorithm (e.g. [3]). Videos may be authored or re-purposed from the segments. For the rest of this paper, a *video* refers to either an original video that has been segmented, or an authored video comprising segments from one or more original videos.

Keywords are associated with the video segments, enabling the video database to be searched like a text database. Keywords are obtained by manually annotating the segments or by temporally matching time-stamped text from transcripts with the video segments. A keyword query produces an ordered list of relevant videos, along with a relevance score for each segment.

CHI 2004, April 24–29, 2004, Vienna, Austria.
ACM 1-58113-703-6/04/0004.



(a) Query (b) Explore (c) Playback

Figure 1. Tasks and transitions: (a) Keyword query and results, (b) Exploration of keyframes from query results, (c) Playback of video segments.

THE MINI-MEDIA SURFER APPLICATION

Now we describe in detail the user interface of MiniMedia Surfer and how it supports the three tasks of query, exploration, and playback of segmented videos. We explain how the application's interaction model with transparent layers supports smooth transitions between the tasks.

The application is designed for mobile devices with a pen stylus. In order for the pen interaction to work across various mobile devices, only a core set of actions is enabled: *tap*, *hold*, and *gesture*. The active pen features (e.g. *hover over*) and the barrel button are not utilized.

The prototype shown in this paper ran on a TabletPC, which provided sufficient performance to handle the digital video. To simulate a small display, a mask was placed over the TabletPC to reduce the viewing area to 320x240 pixels, which is about the size of a PDA screen.

Query by keywords

The first task is a keyword query. A user enters a text string into a text box, and the results of the query are displayed as a list of video titles. This is similar to the popular Google interface. In addition, next to each result is a bar whose height indicates relevance. For this task, a single query layer is visible; it is opaque. See Fig. 1a.

The query results and relevance values are returned by a search engine. The search engine is a separate module hooked up to the MiniMedia Surfer application.

Exploration of search results

The second task is exploration of results. A user investigates promising query results by demanding further detail. The user can select the results one at a time by tapping with the stylus on a result's title. The selected title is highlighted in red, and a second layer appears showing a representative keyframe from the selected video (Fig 1b). Initially, the first frame of the most relevant segment in the video is displayed. The segments of the video, along with their associated keywords are visualized in a timeline at the bottom part of the screen. Each segment has a bar whose height indicates its relevance to the query keyword.

The user can select any segment from the timeline to view its keyframe and keywords. See Fig. 1b and Fig. 2. When there are too many keywords to fit along the width of the screen, the arrows on the keyword panel will be highlighted. The user can scroll the keywords by gesturing left or right on the keyword panel.

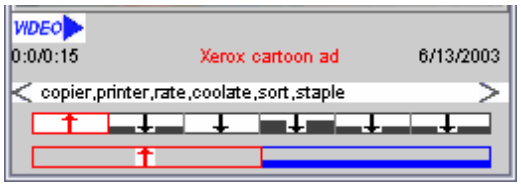


Figure 2. Detail of an expanded segment: The 2nd timeline at the bottom shows the original source video of the expanded segment.

If the video is composed of segments from multiple source videos, an arrow is displayed in the timeline for each segment. Gesturing down on an arrow expands its segment: a second timeline appears for the original source video from which the segment was taken. This is illustrated in Fig. 2. To hide the second timeline, the user makes a gesture along one of its upward pointing arrows.

In transitioning from query to exploration, the transparency values[1] of the query layer and keyframe layer are automatically changed to make it easier to see both layers when they are overlaid on the display. The query layer's alpha value drops from 1.0 to α_1, and the keyframe layer's alpha value rises from 0.0 to α_2; by trial and error we found that $\alpha_1 = \alpha_2 = 0.8$ works well.

The user can adjust the transparency levels of the two layers by making stylus gestures anywhere on the layers. Gesturing to the right increases the opacity of the query layer and gesturing to the left decreases it. Similarly, gesturing up/down will increase/decrease the opacity of the keyframe layer. If the stylus is held down before gesturing, a transparent gradient widget appears. (See Fig. 3).

Figure 3. Transparency gradient widget, gestures, and result of applying a gesture.

This independent adjustment of the two layers is a valuable feature because it is sometimes necessary to adjust the transparency values to reduce occlusion or to see one of the layers better. Also, since mobile devices may be used under different lighting conditions, having greater visual separation between the layers could be more helpful than just uniformly changing the display's brightness or contrast.

Playback of video segments

When the user has found some interesting content, there are several ways to playback the video. Tapping on a selected video title plays that video from the beginning. (Note that double tapping on an unselected video is equivalent to selecting and then playing.) Similarly, tapping on a selected segment plays the video from the beginning of that segment, starting precisely at the displayed keyframe.

By playing the video "in-place" of the keyframe and hiding the query results, a smooth transition occurs from exploration to playback. The visual effect is that the transparent keyframe layer becomes an opaque video layer, and the transparent query layer is faded out. For the playback task, only a single layer is visible. See Fig. 1c.

During playback, a small video controller is activated along the top edge of the video window, and users can stop, pause, or jump to another point on the time slider. Users can also tap on the segments at the bottom of the screen to jump to another part of the video.

[1] The transparency value, also called *alpha value*, is defined as a number between 0.0 and 1.0, with 0.0 being completely transparent to 1.0 being completely opaque.

PROTOTYPE TESTING

We tested the prototype to obtain feedback on the design. Five users participated, three from our lab (none were involved with the project), and two from outside. Two were frequent PDA users.

The users were given three scenarios on browsing videos, and in each scenario they were asked to perform the three tasks of query, exploration, and playback. Exploration is the most complex task, and most of the interaction took place during the exploration of the query results.

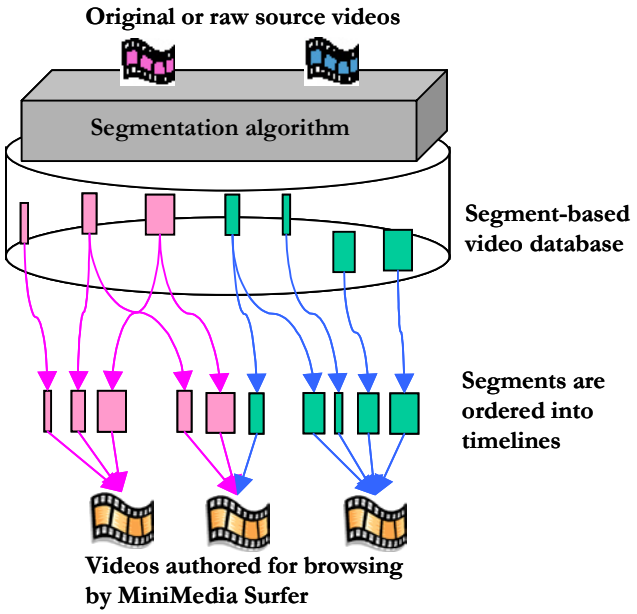


Figure 4. Video data: Each video consists of segments that come from original or raw video footage.

To study the transparency values for optimal viewing of the overlaid query and keyframe layers, the users were asked periodically during the scenarios to set the transparency values for optimal viewing on six pairs of (query layer, keyframe layer). See Fig. 5. The values varied from user to user. Some pairs of alpha values that were desirable are in the neighborhoods of the points $S = \{(1.0, 1.0), (1.0, 0.0), (0.8, 0.7)\}$. We note that the (0.8, 0.7) point is near the (0.8, 0.8) point that we found earlier by trial and error. This point seems to be a good compromise between legibility and occlusion.

One user commented that the optimal values were "always on the right quadrant [sic]" of the gradient widget; from looking at the data in Fig. 5, all points, except for one outlier, have keyframe layer transparency value greater than 0.7. This may be interpreted that during exploration, the user always want to see the keyframe at some fairly high level of visibility, although sometimes the text query results may be obtrusive and need to be made less visible. This data suggest that only some areas on the 2D gradient widget are useful, in particular the keyframe layer should have some minimal transparency value. On the other hand,

allowing the full range of transparency values is simpler and more flexible than over-constraining the values.

Furthermore, two users commented that it was not always necessary to adjust the transparency; one added that it is "sometimes necessary" and the other said it should be an "advanced option". These comments do not detract from the overall design because the gesture technique and pop-up widget for setting transparency values do not normally get in the way of the user or take up valuable screen space.

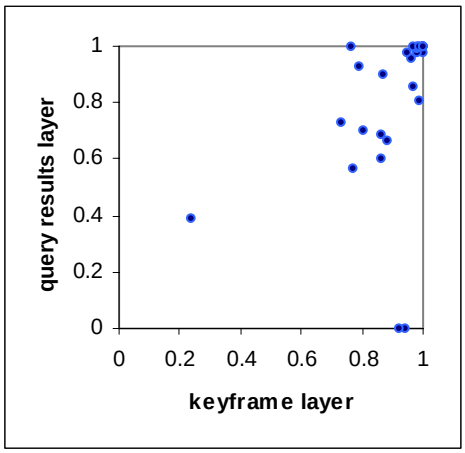


Figure 5. Transparency values set by users for optimal viewing during the five sessions.

The prototype testing also indicated the importance of interaction by gesture on a small device as an alternative to interaction by tapping. Two of five users repeatedly had difficulty tapping on the segments and other small targets. This seemed to be caused by a combination of factors such as parallax of the display, pen calibration, and "shaky hands" of the user. During testing, the device was placed on a table; this problem could have been more pronounced if the user had to hold the device with one hand and the stylus with the other hand. In contrast, we did not observe the users having problems with the gestures.

CONCLUSION & FUTURE WORK

We are encouraged by the effectiveness of the layered interface in MiniMedia Surfer as a way to organize subtasks, to transition smoothly between subtasks, and to deal with limited screen space. We have started looking at related tasks such as reviewing the browsing history; this requires using more than two layers as a way to show history depicted by a stack of keyframes.

Our preliminary prototype testing seems to indicate that gesturing works better than tapping on targets on a small display. This suggests that it might be a good idea to provide gestures to control the video. In early design sketches of MiniMedia Surfer, we had intended to use gestures consistently across the interface for interacting with the video as well as the transparency, segments, and keywords. As a next step for the prototype, we plan to implement and investigate simple gesture commands to play the video segments. More sophisticated techniques for pen-based control of video may also be incorporated (e.g. [9]).

ACKNOWLEDGMENTS

We thank Tohru Fuse for his comments on this project, and the users who helped test the prototype.

REFERENCES

1. Belge, M., Lokuge, I., Rivers, D. Back to the Future: A graphical layering system inspired by transparent paper. *CHI '93 Conference Companion*, pp. 129-130.
2. Bier, E., Stone, M., Pier, K., Buxton, W., DeRose, T. Toolglass and Magic Lenses: The see-through interface. *Proceedings of SIGGRAPH '93*, pp. 73-80.
3. Boreczky, J. and Rowe, L. Comparison of video shot boundary detection techniques. *SPIE Conf. On Storage and Retrieval for Image and Video Databases IV*, San Jose, January 1996, vol. 2670.
4. Girgensohn, A. and Boreczky, J. Time-constrained keyframe selection technique. *Proc. 1999 IEEE Intl. Conf. on Multimedia Computing and Systems*, vol. 1, pp. 756-761.
5. Harrison, B.L. , Ishii, H.,Vicente, K., Buxton, W. Transparent layered user interfaces: An evaluation of a display design space to enhance focused and divided attention. *Proceedings of CHI '95*, pp. 317-324.
6. Kamba, T., Elson, S., Harpold, T., Stamper, T., Sukaviriya, P. Using small screen space more efficiently. *Proceedings of CHI '96*, pp. 383-389.
7. Lyu, M., Sze, S., Yau, E. iVIEW: An Intelligent Video over Internet and Wireless Access System. *The 11th Intl. World Wide Web Conference (WWW2002) Alternate Paper Tracks Proceedings*, http://www2002.org.
8. Maya® software. Alias company, http://www.alias.com.
9. Ramos, G. and Balakrishnan, R. Fluid interaction techniques for the control and annotation of digital video. *Proceedings of UIST '03*, pp. 105-114.
10. Silvers, R. Livemap—A system for viewing multiple transparent and time-varying planes in three dimensional space. *CHI '95 Conference Companion*, pp. 200-201.
11. Uchihashi, S., Foote, J., Girgensohn, A., and Boreczky, J. Video Manga: Generating semantically meaningful video summaries. *Proceedings ACM Multimedia '99*, pp. 383-392.