

# Interactive Audio-Visual Video Browsing

Wolfgang Hürst

Institut für Informatik, Albert-Ludwigs-Universität Freiburg, Germany  
Georges-Köhler-Allee, Building 51, 79110 Freiburg, Germany

huerst@acm.org

## ABSTRACT

We present the AV-ZoomSlider interface for video browsing. It complements existing approaches, such as storyboards and video skims by enabling users to interactively navigate along the time line of a video file. Our solution smoothly integrates position- and speed-based navigation concepts and provides synchronized audio-visual feedback during scrolling when applicable.

## Categories and Subject Descriptors

H.5.2 [Information Interfaces and Presentation (e.g. HCI)]:  
User Interfaces – *Graphical user interfaces (GUI)*

## General Terms

Design, Human Factors.

## Keywords

Video browsing, multimedia user interfaces, interactivity.

## 1. INTRODUCTION

One of the most common approaches for video browsing is to automatically extract key frames or “key clips” based on, for example, syntactic information such as scene changes or camera cuts. This information is then presented, for example, as static or dynamic storyboards, video summaries or skims which can easily be browsed by the users [1]. We call these approaches *system-controlled* because the system preprocesses the signal and represents it in a suitable way for browsing. Such techniques are often complemented by approaches where the users themselves browse a file’s content interactively, for example, when searching for particular events which are of more personal but less general interest (and therefore often not represented in a storyboard or summary). The most straightforward way to offer such an interactive, *user-controlled* browsing is to include some sort of controller into the interface enabling users to modify replay speed at various levels. In the following, we refer to these approaches as **speed-based** navigation or browsing. Another possibility for interactive video browsing is the use of a time-based slider in combination with real-time random access to the respective video file. Because of the random access, any visual change can be seen immediately while a user moves the slider’s thumb along the timeline thus enabling very flexible data browsing. In the following, we refer to such approaches as **position-based**, since they allow users to directly control the position which is currently displayed (and browse the respective content). A good system

design should include both speed- and position-based approaches, because they normally complement each other very well. For example, speed-based techniques are usually the mode of choice for skimming larger parts of a file at a constant speed, whereas position-based approaches are generally more suited if navigation on a finer level and exact positioning are required.

In this paper, we introduce the AV-ZoomSlider interface which is an extension of our previous work on position-based browsing that we summarize in Section 2.1. In Section 2.2, we describe how speed-based skimming can be integrated seamlessly into the ZoomSlider thus complementing the existing interface in a useful way. In Section 3, we introduce the ability for synchronized audio-visual data browsing. Section 4 presents a discussion of the AV-ZoomSlider’s usability and gives an outlook to future work.

## 2. BROWSING VISUAL DATA STREAMS

One of the main advantages of position-based browsing is the ability to quickly change between different levels of granularity when navigating a file. For example, by quickly moving a slider thumb along the timeline, users can easily skim larger parts of a file to get a rough overview of its content. Once a particular part of interest is located, users can immediately stop (by pausing their movements) and examine the respective scene in more detail (by slowly moving the slider thumb back and forth). However, in the latter case, a *scaling-problem* exists: Since sliders are restricted in length by the size of the respective player window, the mapping of each random frame of a long video to a position on the slider is no one-to-one and onto function. Hence, even the smallest possible move of the slider thumb (i.e. one pixel on the screen) can result in a large jump in the file. The resulting jerky visual feedback is often considered irritating by the users and, in the worst case, might cause that particular information is skipped completely. A common way to deal with this scaling problem is to enable users to rescale the slider’s granularity by zooming into the slider’s scale. Additional interface elements, such as zoom buttons, allow users to set the scale to the level of detail which seems most appropriate for a particular situation. Such approaches usually work well, for example, in video editing [2] where users have to handle different tasks and continuously switch between a variety of widgets anyhow. However, in case of browsing, for example, when searching for information, the required switches between different interface elements interrupt the skimming process and are therefore harder to handle by the users.

### 2.1 Position-based browsing

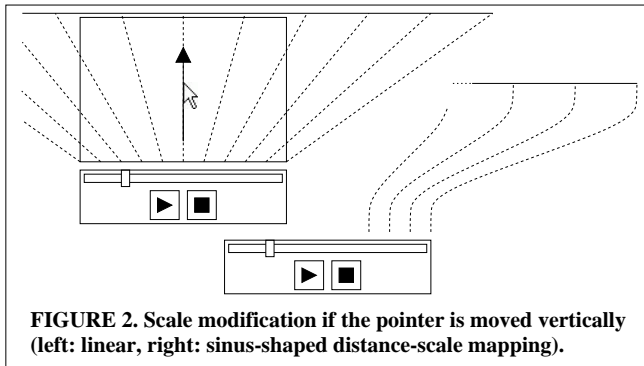
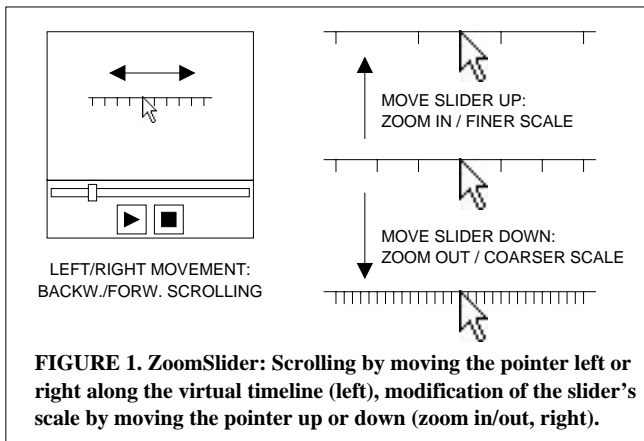
Our ZoomSlider [3, 4] solves the scaling-problem by integrating both interactions, i.e. zooming and scrolling, into one interface by using the second dimension which is orthogonal to the slider’s orientation: If users click anywhere on the player window and move to the left or right, they navigate through the file backward or forward along a virtual, horizontally mounted timeline in the same way as if they had clicked on the original slider bar (Fig. 1,

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

MM’06, October 23–27, 2006, Santa Barbara, California, USA.

Copyright 2006 ACM 1-59593-447-2/06/0010...\$5.00.

left). If they move the mouse pointer vertically (i.e. orthogonal to the orientation of the slider), the scale of this virtual timeline is modified – moving the pointer upwards results in a finer scale, downwards movements cause a zoom-out (Fig. 1, right). The top of the player window is associated with the finest possible scale (i.e. one pixel on the screen corresponds to one frame in the video), while the scale at the bottom, next to the “real” slider, is defined through the length of the document and the width of the player window (i.e. it is the same as with the original slider). In between, the scale is adapted accordingly. Figure 2 illustrates the scales of the virtual timelines if a user moves strictly vertically.



While the basic idea for the ZoomSlider seems simple, there are a lot of degrees of freedom when it comes to its actual realization. In the following, we summarize the most important and crucial parameters. A detailed discussion along with experiments and evaluation results can be found in [3] where we presented an initial version of the original, purely position-based ZoomSlider.

In Figure 2 (left), the resolution of the virtual timeline’s scale changes linearly if the pointer is moved up or down. In practice, it turned out that such an implementation leads to the (subjective) impression of an abrupt scale change at the beginning that often irritates the users. Based on several informal studies, we therefore propose a sinus-shaped mapping of the scale change as illustrated in Fig. 2 (right). Such an implementation has several advantages: (a) the two most important scales (i.e. the original slider’s scale and the finest possible scale) are represented through a larger area, (b) the transition from these scales to the ones in between does not take place abruptly but happens smoothly, and (c) in the middle, a nearly linear modification of the scale takes place (cf. Fig. 2).

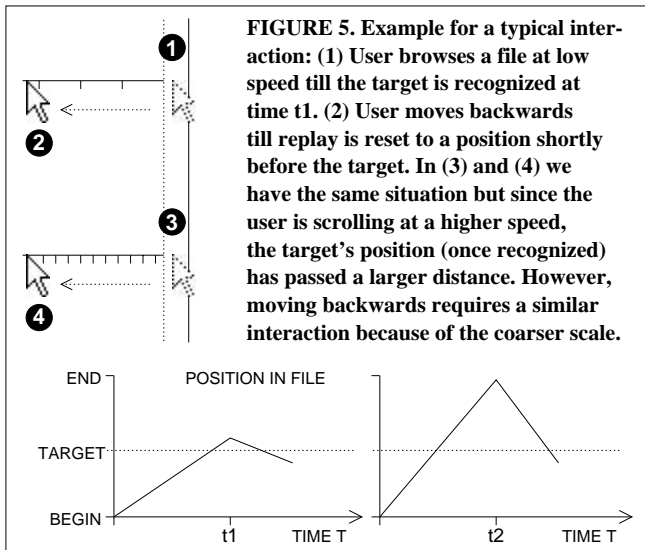
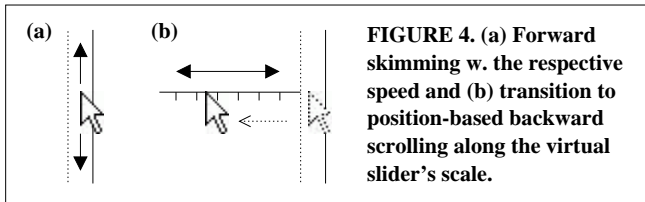
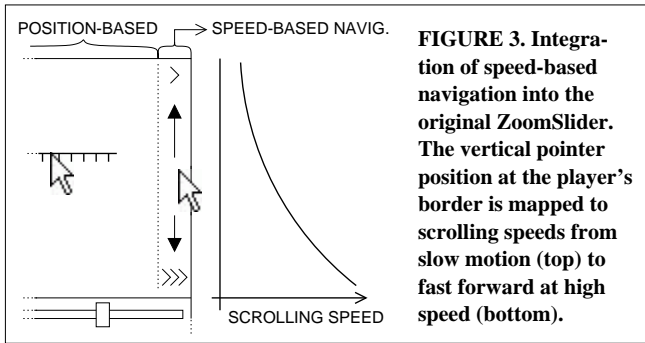
Having to move the pointer up or down in order to get to a finer or coarser scale, respectively, can be frustrating. In addition, even when the user’s aim is to strictly move the pointer vertically, often small variations in the horizontal direction occur, what proves to be a critical issue in the lower regions of the player window (i.e. the area with a coarse scale resolution) due to the scaling problem. In order to deal with these issues, we propose an implementation where users can not only modify the scale resolution by moving the pointer up or down continuously but also by directly clicking anywhere on the player window, in which case the respective scale (such as illustrated in Fig. 2) is applied. It should be noted that such a realization is only possible because of the clear association of the window borders with the minimum and maximum scale resolution, respectively, and distinguishes our approach from related work on 2D scale modification (e.g. [5, 6]).

## 2.2 Integration of speed-based browsing

Sliders for position-based navigation along the timeline provide an intuitive, easy to use, very powerful and flexible way for browsing. However, there are situations in which speed-based skimming seems to be the better choice. For example, if a user wants to scan larger parts of a file in search for some information, it might be more pleasant to just increase replay speed and watch the accelerated replay till a scene of particular interest appears. On the other hand, if a user wants to find a specific event or object within that scene, more interactivity might be required, such as going back and forth a few seconds in order to do an exact positioning. Such interaction is generally hard to do with pure speed-based navigation but much easier to handle with a position-based, slider-like interface. Hence, speed- and position-based navigation generally complement each other very well, and an optimal design should offer both kinds of interaction. However, most common video browsing systems provide these two interaction styles through separate widgets. They require from the users to continuously switch between different interface elements thus interrupting the overall skimming process. In the following, we present an extension of our ZoomSlider interface which smoothly integrates speed-based browsing into the existing, position-based navigation techniques.

In this advanced ZoomSlider we added speed-based browsing to the borders of the player window in the following way: If a user reaches the right border of the player window, scrolling switches from the position-based navigation described above to a speed-based skimming where the file is replayed in a typical fast-forward or slow motion fashion. Replay speed is set against the vertical position of the mouse pointer: Beginning with the lowest scrolling speed of 0.5 times normal replay at the upper part of the player window, replay speed continuously increases the more the user moves the mouse pointer down, such as illustrated in Figure 3. At the left border of the player window, the same speed-based behavior is realized but here for backwards replay.

In the following, we describe the functionality and advantages of this new ZoomSlider design with a simple example. Assume a typical situation where a user is looking for an event of particular interest which is located somewhere in a video recording. If a user has no information about where the target position might be located, one good and common approach is to start skimming the file at a higher speed in order to roughly find an area of likely interest. Once the respective part of the file is located, the user might want to switch scrolling mode from speed- to position-based navigation in order to examine the respective part in more



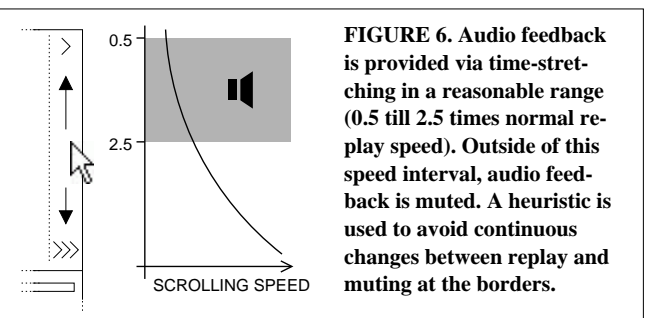
detail by going back and forth, by navigating at different granularity levels, etc. Our new ZoomSlider interface supports this kind of interaction very well. First, a user can skim the file at different replay speeds using the speed-bar located at the right border of the player window (Fig. 4a). Speed can be modified by moving the pointer up or down similar to a (normally horizontally mounted) controller-like interface which is commonly used to provide such functionality. Once a particular part of interest is found, the user can switch to position-based navigation by simply moving the pointer to the left (Fig. 4b). Because of the way in which the ZoomSlider is designed, such a movement automatically results in a change of the scrolling direction, which is the most natural and likely interaction in such a situation (since the user already passed the relevant part once he / she has seen it and now wants to go back to re-view it and explore it in more detail). The amount a user normally wants to go back in the file is influenced by the previous scrolling speed: If speed was higher, users generally want to go back a larger interval (since the information that just passed is further away). If speed was lower, normally a much smaller re-set is necessary. Again, this situation is supported very well by the ZoomSlider because the scale of the

virtual timeline and the respective replay speed are harmonized with each other insofar that from a higher replay speed users turn into a coarser scale (and vice versa, see example in Fig. 5) thus resulting in a seamless transition and a higher likelihood that the user quickly reaches the target position. In a similar way, we can argue that there are situations where a smooth transition from position-based to speed-based navigation is desired and greatly supported by the ZoomSlider interface.

### 3. AUDIO-VISUAL BROWSING

The previously introduced ZoomSlider interface enables users to browse continuous visual data streams in a flexible and powerful yet intuitive way by smoothly integrating speed- and position-based navigation. However, there are also situations where the information a user is looking for is not located in the visual part of the data but within the synchronized audio track. Typical examples are talk shows, where the visual signal just represents a talking head but the actual information can only be perceived by skimming the audio stream. While the usefulness and benefit of skimming a video's audio track has been verified in the past [7, 8], common interfaces normally neither integrate audio-visual browsing seamlessly into one interface nor give users adequate control and flexibility to synchronously skim the acoustic and visual content at different granularity levels. In the following, we describe another extension to our interface which combines visual browsing with time-synchronized audio skimming thus introducing the final AV-ZoomSlider.

Intelligible audio feedback can only be provided if a set of consecutive samples is replayed (unlike video where single frames can be represented in complete isolation). Therefore, speed-based navigation methods are the most obvious and natural way for providing interactive, user-controlled audio browsing. Previous work on audio skimming (e.g. [7, 8]) showed that for speech recordings replay speeds of up to 2.5 times normal replay are still useful for browsing if signal processing is used to preserve the original pitch, thus avoiding typical cartoon character-like voices. Figure 6 illustrates how such time-stretched audio feedback is integrated into the AV-ZoomSlider.



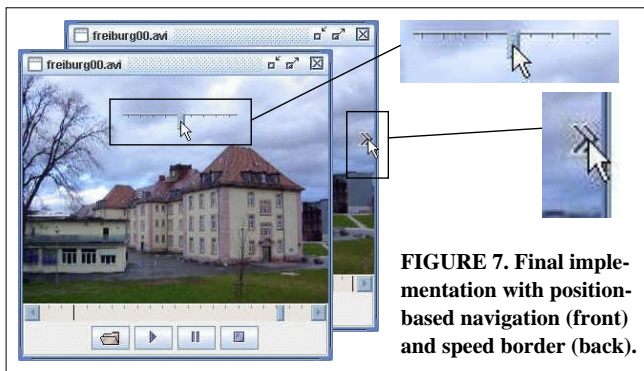
The audio skimming functionality realized by this extended AV-ZoomSlider is a simple time-stretching mechanism offered by many other audio browsing interfaces. However, the important difference to common approaches is its smooth integration into the overall interaction process. Generally, visual information is much easier to scan than acoustic signals, especially at higher skimming rates. Hence, people often use visual skimming to get a rough overview or to quickly identify a part that might be interesting and only fall back on audio skimming to further evaluate this particular scene. The AV-ZoomSlider perfectly

supports such scenarios because audio feedback is smoothly integrated into the visual browsing process. For example, users can skim the visual part of a file at high speed to find all positions where a particular talk show guest is speaking. If such an event is discovered, they can slow down, thus automatically enabling audio feedback to find, for example, a part where the person is making a specific statement. If exact positioning is needed, for example because they want to re-set replay to the exact frame where the statement begins, they can switch to position-based navigation by simply moving the pointer to the left.

#### 4. DISCUSSION AND USABILITY

In the preceding sections we provided intuitive arguments and examples for the usefulness of the introduced AV-ZoomSlider interface. The proposed design integrates several functionalities smoothly into one interface. Hence, it is easy to create cases where the ZoomSlider outperforms, for example, standard sliders (because they do not offer browsing on different granularity levels), pure position- or speed-based approaches, audio skimming techniques that do not integrate fast visual browsing, etc. Therefore, the most interesting and critical question is if the provided functionality is integrated in a way that does not significantly increase complexity but still offers an easy and intuitive interaction experience to the average users.

In order to verify this, we performed a heuristic evaluation (heuristic evaluation is an established approach for user interface evaluation in early design stages that is usually performed with five experts who serve as sample users and evaluate usability based on a given set of heuristics [9]). In addition, we made some informal tests with twelve users who never saw the AV-ZoomSlider before and evaluated if they are able to understand and handle the provided browsing functionality. Finally, those twelve users had to perform some search tasks on different granularity levels (such as finding a particular scene, a particular frame, etc.) and in different media types (such as finding positions where a specific person says a particular sentence). Our evaluations confirmed that despite its increased complexity users are able to handle the AV-ZoomSlider very well, even without significant training. The resulting performance in the search tasks does not only suggest a high usability but gave initial evidence that users will also be able to find relevant information faster. Detailed, quantitative experiments are part of our future work.



**FIGURE 7. Final implementation with position-based navigation (front) and speed border (back).**

Figure 7 illustrates our final implementation of the AV-ZoomSlider interface which resulted from several evaluations with alternative visualizations. A reasonable visualization supports users to understand and distinguish the various

functionalities offered by our system and therefore, is essential for the overall usability. In the on screen visualization near the pointer, the actual scale resolution is presented and continuously adapted if the pointer is moved vertically. In the original slider widget at the bottom of the player window, the slider's scale is adapted accordingly when the pointer is moved up or down. This way, the overall position within the file is always clearly indicated. In addition, the scale is continuously updated when a user moves along the virtual timeline, thus creating the illusion of a slowly moving slider thumb even in fine granular navigation, as proposed by [10].

In the same way as we argued that speed/position-based and audio/visual-based navigation complement each other, there are scenarios where system-controlled approaches are preferred over the user-controlled-techniques presented here (and vice versa). Hence, our current and future work addresses the question how typical interaction scenarios for system-controlled approaches relate to browsing behaviors performed with the AV-ZoomSlider and how both concepts should therefore be integrated into one single interface. In addition, first experiments with the AV-ZoomSlider on a pen-based Tablet PC have been very promising. Based on this experience, we assume that the AV-ZoomSlider might be a very promising approach for browsing video on pen-based mobile devices, such as PDAs.

#### 5. REFERENCES

- [1] Y. Van Houten: *A framework for video content browsing*. Enschede, Telematica Institute, June 2002. Available at <http://doc.telin.nl/dscgi/ds.py/ViewProps/File-12409>
- [2] J. Casares, B.A. Myers, A.C. Long, R. Bhatnagar, S.M. Stevens, L. Dabbish, D. Yocum, A. Corbett: *Simplifying Video Editing Using Metadata. Proceedings of Designing Interactive Systems (DIS 2002)*, London, UK, 2002
- [3] W. Hürst, P. Jarvers: *Interactive, Dynamic Video Browsing with the ZoomSlider Interface. Proceedings of the IEEE International Conference on Multimedia & Expo (ICME 2005)*, Amsterdam, The Netherlands, 2005
- [4] W. Hürst, G. Götz, P. Jarvers: *Advanced User Interfaces for Dynamic Video Browsing. Proceedings of ACM Multimedia 2004*, New York City, NY, USA, 2004
- [5] G. Ramos, R. Balakrishnan: *Fluid interaction techniques for the control and annotation of digital video. Proceedings of ACM UIST 2003*, Vancouver, BC, Canada, 2003
- [6] C. Ahlberg, B. Shneiderman: *The Alphalider: A compact and rapid selector. Proceedings of ACM CHI 1994*, Boston, MA, USA
- [7] A. Amir, D. Ponceleon, B. Blanchard, D. Petkovic, S. Srinivasan, G. Cohen: *Using Audio Time Scale Modification for Video Browsing. Proceedings of the 33<sup>rd</sup> Hawaii International Conference on System Sciences (HICCS00)*, Maui, HI, USA, 2000
- [8] L. He, A. Gupta: *Exploring Benefits of Non-Linear Time Compression. Proceedings of ACM Multimedia 2001*, Ottawa, Canada, 2001
- [9] J. Nielsen, R.L. Mack (eds.) *Usability Inspection Methods*, J. Wiley & Sons, New York, NY, USA, 1994
- [10] Y. Ayatsuka, J. Rekimoto, S. Matsuoka: *Popup vernier: A tool for sub-pixel-pitch dragging with smooth mode transition. Proceedings of ACM UIST 1998*, San Francisco, CA, USA, 1998