

Interactive Video Browsing on Mobile Devices

Wolfgang Hürst
Albert-Ludwigs-University
Georges-Köhler-Allee, Building 51
79110 Freiburg, Germany
huerst@acm.org

Georg Götz
Albert-Ludwigs-University
Georges-Köhler-Allee, Building 51
79110 Freiburg, Germany
mail@georg-goetz.de

Martina Welte
Albert-Ludwigs-University
Georges-Köhler-Allee, Building 51
79110 Freiburg, Germany
welte@informatik.uni-freiburg.de

ABSTRACT

Today, videos can be replayed on modern handheld devices, such as multimedia cellphones and personal digital assistants (PDAs), due to significant improvements in their processing power. However, screen size remains a limiting resource making it hard, if not impossible to adapt common approaches for video browsing to such mobile devices. In this paper we propose a new interface for the pen-based navigation of videos on PDAs and multimedia cellphones. Our solution – called the *MobileZoomSlider* – enables users to intuitively skim a video along the timeline on different granularity levels. In addition, it allows for continuous manipulation of replay speed for browsing purposes. Both interaction concepts are seamlessly integrated into the overall interface, thus taking optimum advantage of the limited screen space. Our claims are verified with a first evaluation which proves the suitability of the overall concept.

Categories and Subject Descriptors

H.5.2 [Information Interfaces and Presentation (e.g. HCI)]: User Interfaces – *Graphical user interfaces (GUI), input devices and strategies, interaction styles, screen design*

General Terms

Design, Human Factors.

Keywords

Handheld devices, pen-based computing, mobile computing, interface design, interaction, video browsing, navigation.

1. INTRODUCTION

A lot of newer handheld devices, such as personal digital assistants (PDAs), mobile media players, or mobile multimedia phones, are able to seamlessly replay even longer video files (such as whole movies) in a high quality. However, people use these devices not only to watch movies and TV shows, but to access a large variety of different contents: Private videos recorded during a vacation trip, short video clips captured with their cellphones, movie trailers, sports and news clips downloaded from the internet or delivered directly to the mobile device via podcasting, and so

on. The spectrum of data varies from very long movies to rather short video clips as well as from low quality, amateur clips to high-quality, professionally produced files.

The interface for replaying those files on the mobile devices is generally based on the traditional tape recorder metaphor which is normally well suited for situations where rather long videos are watched continuously. However, due to the large variety in different data, we can assume that users of mobile video often want to interfere with normal replay, for example, to skip single clips or parts of lower interest within a longer file. They want to reset replay to a position of higher interest, pause replay in a private video if a known person appears, and so on. In addition, it is likely that people often tend to watch for rather short periods of time when being “on their way”, thus increasing the need for navigation in longer video files. These intuitive arguments are supported by a recent study about the everyday usage of mobile video [14]. The authors do not only identify different usages of video on mobile devices but also analyze the social motivations and values underpinning typical user behavior. Their study confirms that people in deed access a variety of files of different contents, sizes, etc. In addition, they identify different scenarios requiring intensive interaction. For example, they observed a situation where several people were sitting around a PlayStation Portable (PSP®) in order to show each other their favorite scenes from a popular movie and watch them together.

Lots of research on video browsing has been done in the last couple of years because of the general need for advanced navigation and interaction techniques in relation to digital video (see [21] for a good overview). Unfortunately, most of this work has been restricted to video replay on desktop PCs and laptops. However, there are significant differences between such traditional computers and handheld, mobile devices, i.e.:

- *Performance.* Despite recent improvements, performance remains a big issue for mobile devices: Low battery life prohibits high clock rates; design and size of the devices make heat transmission difficult; limitations exist, e.g., in the achievable frame rate due to restricted processor power and memory; etc.
- *Input devices.* Keyboard, mouse, and touchpad are predominant input devices on PCs and laptops. In contrast to this, handheld devices are usually operated by much fewer buttons which are laid out in a special way (e.g. cell-phones), a pen (e.g. PDAs and some cell-phones), or even fingers (e.g. the new Apple iPhone).
- *Screen size.* The terms “mobile” and “handheld” imply that there is (and always will be) a natural limit for the screen size, i.e. the area in which to represent the content and additional interface elements for direct interaction.

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.

ACM Multimedia 2007, September 24–29, 2007, Augsburg, Germany.
Copyright 2004 ACM 1-58113-000-0/00/0004...\$5.00.

Because of these differences, most of the commonly used approaches for video browsing can not be applied to handheld devices. In this paper, we present the *MobileZoomSlider*, a new interface for interactive video browsing on pen-based, handheld devices. First, we review related work on video browsing in general and discuss it in relation to video browsing on mobile devices (Section 2). Then, we introduce our interface, compare it to related approaches, and comment on the actual implementation (Section 3). Finally, we present a user study which proves the feasibility and usefulness of our design (Section 4) and gives hints for further developments and future work (Section 5).

2. RELATED WORK

Many useful approaches for video browsing which have been introduced in the past can hardly be applied to mobile video due to the limited screen size. For example, storyboards – i.e. thumbnail representations of single scenes of a video [6] – require too much screen space in order to be useful on a small display. Even on larger screens there often exists an “overview-detail trade-off” [22], i.e. if too many scenes are shown, the content of each one can hardly be recognized because of the reduced thumbnail size, and if fewer scenes are represented through larger thumbnails, significant parts of the video are not visualized. Similar arguments can be given for the limited usability of “dynamic displays” [21] such as motion storyboards [20] and video skims [23].

Automatically generated summaries and trailers can be replayed on mobile devices and are useful in some situations, for example to get a quick overview of a file. However, to identify a particular part of interest, for example the frames representing a known person in a private video recording, further approaches are needed. [12] introduces a special way for large image browsing on PDAs. Here, important parts of an image, such as people’s faces, are automatically identified and a “trail” through a zoomed in version of the image is created. The Diver project [16] uses a somehow related technique for panoramic video recordings. Again, such approaches are indeed useful for video on mobile devices, but do not cover any possible search or navigation task users might have.

One of the main issues making video browsing much harder than, for example, text or image browsing, is the fact that video has a temporal component, i.e. the visually presented information from a video is not static but changes over time. Therefore, another, more interactive approach for video browsing is to provide interfaces which give the user more control over which part of a file is replayed at what time. One obvious way to do this is to enable users to modify the replay speed: Faster replay can be used, for example, to get a quick overview of a file (e.g. in order to find scenes where particular events take place). Slower replay is useful for a more detailed navigation (e.g. to reset replay to the beginning of a particular scene or to find exactly those frames where a specific person appears in). In the following, we will refer to this kind of video browsing as *speed-based* navigation. In theory, such approaches should work on mobile devices as well. In practice, however, performance can become an issue, since there might be a maximum limit for faster replay. Another question is how the interface should be designed in order to offer such functionality to the user. Normally (i.e. for media players on desktop PCs), common widgets such as different buttons, controller wheels, or a slider are provided to allow users to set

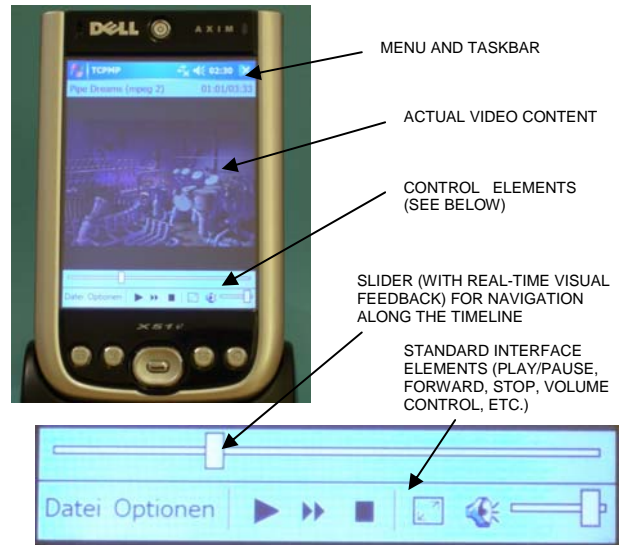


Figure 1. Standard video player interface on a PDA.

replay speed to a specific value. This can be a problem on a mobile device where screen size is limited (cf. Fig. 1): Standard media player interface elements, such as buttons for play, pause, and stop, already use up a large part of the screen. In addition, a timeline-based slider is often provided for navigation (cf. next paragraph) as well as indication of file length and relative replay position. Adding other interface elements, such as a slider for manipulation of replay speed, quickly fills the screen and leaves less space for the representation of the actual content.

A hardware-based approach, i.e. using the keys and even joysticks often found on modern mobile devices, is no solution for this problem as well, because a key-based interaction can never implement the variety and flexibility provided by direct manipulation interfaces such as sliders for continuous speed modification.

Another approach for interactive video browsing is what we call *position-based* navigation: Here, visual feedback is provided in real-time while a user moves a timeline-based slider representing the length of the video (cf. Fig. 1). Such an interface has proven to be very useful for video browsing, because it enables users to quickly skim the content of a file, easily skip parts of minor interest, identify relevant portions of the file, quickly slow down and change scrolling direction, etc.

However, the limited screen space problem, which we already described before in relation to speed-based browsing, exists here as well. In addition, the common design of sliders can easily lead to input errors: Often, sliders are designed in a way that clicking somewhere next to the thumb on the timeline results in a jump of the slider’s thumb (cf. Fig. 2). This functionality can be very useful to quickly jump to, for example, the last quarter of a file. However, if the slider interface is rather small, then the slider’s thumb is hard to target and a user might often accidentally click next to the thumb when trying to grab it, thus resulting in an unwanted change of the current position in the file. Hence, there is a general trade-off for the design of widgets on mobile devices: Designing them too small makes them harder to target and in some cases can easily lead to erroneous input. However, larger

widget versions use up a lot of screen space which is a critical resource on handheld devices.

A further significant disadvantage (esp. for small screen sizes) is that sliders do not scale to large document sizes: The length of a slider is restricted by the size of the player window and the screen resolution. Hence, if a document is rather long, not every position in the file can be mapped onto a position on the slider's timeline. The result is a jerky visual feedback during scrolling, which is usually considered as very disturbing. In addition and even more critical, single parts of the file (in the worst case complete scenes) can not be accessed directly but are skipped completely. Common approaches to deal with this problem by, for example, providing more buttons to modify the scale resolution of the slider or additional timelines with different scales [18] are generally not suitable for mobile devices due to space limitations, as already discussed above.

Since the scaling problem also appears for static data (e.g. a long text and a scrollbar) and long video files on desktop PCs, there are several approaches that try to solve it by introducing new widgets or extending existing ones. We will discuss some of the most relevant approaches for the scaling problem later when we compare them to our solution. Unfortunately, most of these solutions can not be easily transferred from desktop PCs to handheld devices. To the best of our knowledge, so far no approach exists that deals with this problem in relation to video browsing on small screen sizes and pen-based input devices.

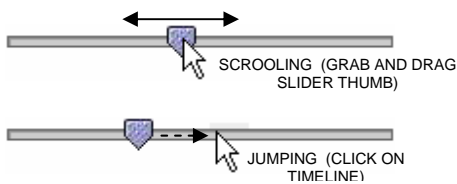


Figure 2. Common slider functionalities.

3. THE MOBILEZOOMSLIDER

In the following, we introduce the basic idea of our MobileZoomSlider for pen-based video browsing on handheld devices (3.1), compare it to related approaches for solving the scaling problem (3.2), and finally give some details about its implementation (3.3).

3.1 The ZoomSlider Interface

The basic idea of the ZoomSlider is to provide scrolling functionality to the user without the need for particular widgets (e.g. timeline-based sliders) but by just clicking on the screen: A click anywhere on the video followed by a left or right movement of the pen results in a backward or forward navigation in the file (cf. Fig. 3a). In order to enable users to navigate through the file at different granularity levels, the resolution of the scale used for scrolling depends on the vertical position of the pen: If a user moves the pen horizontally at the bottom of the screen, the same (coarse) scale is used as in the timeline of the original slider. If one clicks at the upper part of the display, the finest possible scale is used (i.e., one pixel on the screen corresponds to one frame in the video). In between, the scale is adapted according to the vertical position of the pen (cf. Fig. 3b).

The main advantages of this realization are that a) there is no need to “catch” a small, moving target (i.e. the slider thumb) but interaction starts easily by clicking anywhere on the screen, and b) there is no need to explicitly set the scale for fine or coarse scrolling because we use the second dimension to manipulate this parameter. The latter issue limits the need to present additional interface elements such as buttons to zoom into the scale or a second slider bar with a different scale resolution. In addition, it reduces the amount of interaction required by the user – an issue which is considered to be a general design guideline for interacting with mobile devices [7].

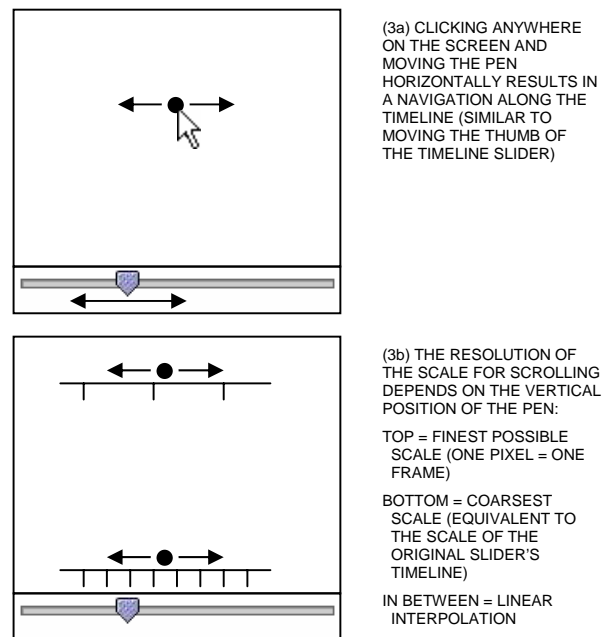


Figure 3. MobileZoomSlider: Position-based navigation.

Position-based browsing is very useful, if a user wants or needs to skim a file at different granularity levels, for example quickly on a coarser scale to skip a part of minor interest and then slowly and in varying directions on a finer scale in order to closer examine a particular scene or to reset replay to a specific frame. However, for longer scrolling at a fixed scale, manipulation of replay speed often seems to be the better interaction mode. In addition, when moving along the timeline at a fine scale in the implementation described before, the screen borders are quickly reached thus requiring the user to reposition the pen. For this reason, we introduce a “speed-border” to our ZoomSlider interface: If the user reaches the right border of the window (or clicks there directly), speed-based navigation in the file is evoked (cf. Fig. 4a). Similar to the scale for position-based navigation (cf. above and Fig. 3b), replay speed depends on the vertical position of the pen: Faster replay is realized at the bottom of the window, whereas the top realizes a slow motion. In between, an interpolation of the replay speed is done (cf. Fig. 4b). One of the main advantages of this speed-border is that, again, we are able to provide this extended browsing functionality without the need to display any additional widgets on the screen.

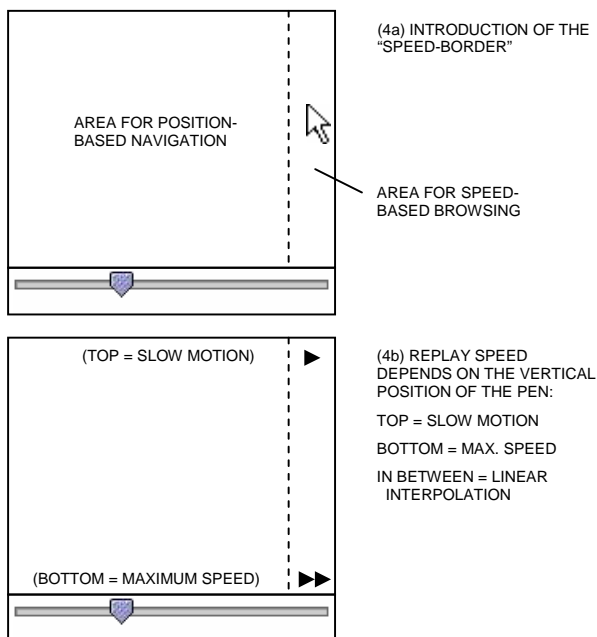


Figure 4. MobileZoomSlider: Speed-based navigation.

Another advantage of the speed-border is that it smoothly integrates speed-based with position-based navigation: Since the vertical position of the pen determines scale-resolution and scrolling speed, respectively, slow and fast replay are correlated to fine and coarse scale resolutions if a user moves the pen horizontally from the position-based to the speed-based navigation region (and vice versa). In the original implementation, we manually set a scale and replay speed range and mapped them to the respective vertical position on the screen. In a further extension of the interface, we internally calculate the speed at which a user navigates along the timeline during position-based scrolling and, once the user moves from the position- to the speed-based region of the player window, we use a speed value that matches the current scrolling speed. Then, users can increase or decrease this value by moving the pen down or upwards, respectively, within the speed-border region. The need to switch between these two interaction modes is actually quite common in search tasks: First, users skim a file at a specific replay rate to, for example, locate a part of particular interest. Then they switch to position-based navigation in order to examine the respective scene in more detail and reset replay to a particular frame. In our implementation, such a transition from one interaction mode to another can be done very easily and intuitively without the need to manipulate any separate user interface elements or widgets

Hence, our implementation integrates both interaction concepts in a clever way and provides a large range of interaction functionalities while at the same time avoiding the need to display various interface elements. In addition, both ideas – offering position-based scrolling at different granularity levels by just moving the pen horizontally on the screen as well as providing speed-based navigation by moving the pen up and down at the window border – are actually rather simple approaches and therefore should be intuitive and easy to understand and operate by the users. In the following, we first compare our design to existing approaches considering their feasibility on pen-based,

handheld devices. Then, we discuss our own implementation on a PDA, describing the actual realization including performance issues and visualization of the interface elements.

3.2 Comparison with Related Approaches

From an abstract level, the scaling problem related to sliders and long videos also appears when scrollbars are used to navigate in static data (e.g. long text files or large images). Not surprisingly, different approaches exist to deal with this problem in both cases, i.e. navigation within static data as well as for video browsing. However, many of these approaches are either not suitable for pen-based handheld devices or do not offer the same range of functionality as our MobileZoomSlider interface.

Using additional widgets such as buttons or supplementary sliders to enable users to modify the granularity of the timeline's scale or replay speed is generally not feasible here because of the limited screen size, as already discussed above. Another approach often used is to do a different mapping between the mouse movements done by the user and the respective navigation within the document. For example, if the right mouse button is pressed a finer scale can be used [3]. Obviously, such approaches are not feasible for pen-based computing.

The so called Alphaslider introduced by Ahlberg and Shneiderman [1] is one of the most well-known approaches to deal with the scaling problem for static, time-independent data. Here, the slider thumb is divided into three different areas, each of which allows users to scroll through the respective file at different granularity levels. Unfortunately, this approach is unsuitable for our case since a) internally it relies on a different mapping of mouse movements for slower navigation and therefore can not be applied to pen-based interaction and b) selecting small targets (i.e. something that is even smaller than a slider thumb) with a pen on a very small device is very critical as already discussed above.

In [1], Ahlberg and Shneiderman also evaluated an alternative version of the Alphaslider where the different granularity levels of the scale are not selected directly but instead modified by moving the mouse orthogonal to the slider's orientation. This is comparable to our approach in the position-based navigation area insofar as the second dimension is used to modify the scale. However, Ahlberg and Shneiderman's approach (as well as several other approaches which take advantage of the second dimension for scale modification, e.g. [13, 17]) requires users to initially click on the slider and subsequently modify the scale by keeping the mouse or pen pressed while moving up or down. Ahlberg and Shneiderman [1] evaluated that such an approach usually fails in practice because users often release the mouse buttons during movements unintentionally. Even if this happens for just a few milliseconds, it always requires the users to go back to the initial slider interface and do a re-scaling. Our own experiments showed that this problem is even more critical when a pen and a PDA are used as input device.

In the newly introduced Apple iPhone [2] a special mechanism is used to navigate through longer lists of text which to some degree is comparable to the "elastic slider" approach we introduced in [8] for video browsing. Our approach uses the so-called rubber-band metaphor to enable users to skim through a file at different speeds, whereas in case of the iPhone the user "pushes" the document or list of items in one direction. Depending on the momentum of the input, scrolling slows down faster or slower. Both cases do not realize a real position-based navigation but allow users to navigate

a file by manipulation of the speed in which the text moves or the video is replayed, respectively. However, the evaluation we present in Section 4 confirms our argument that position-based navigation is very important, especially for video browsing.

The only approach we are aware of which combines both interaction modes, i.e. speed- and position-based navigation, is the PVslider introduced in [17]. It is also the most similar one to our MobileZoomSlider interface but significantly differs in some important aspects. Figure 5 shows our re-implementation of the PVslider interface which we did in order to compare its performance and usability to our approach. The basic idea of the PVslider is also to click anywhere on the screen to initiate position-based scrolling. If the pointer is moved up or down, the scale of the timeline (which initially has the finest possible resolution, i.e. one pixel is associated with one frame in the file) gets coarser. In contrast to our ZoomSlider, accessing different granularity levels by directly clicking in a certain area of the screen is not supported. The resulting need to continuously keep the pen or mouse button pressed while modifying the scale resolution is critical in practice, as already discussed above. In addition, one always has to start with the finest possible scale, even if just a fast navigation along the coarser scale is required by the user. This is the direct opposite to a typical search task as described by [3] where users generally start with a coarser scale and subsequently want to switch to a finer scale resolution. Similar to the ZoomSlider, scrolling switches to speed-based navigation after navigating along the timeline for a while. However, in contrast to our approach, the area for speed-based navigation is not associated with the window border but depends on a fixed value for the distance between the initial clicking position and actual position of the pen. Hence, position-based navigation is restricted to a rather short segment whereas the ZoomSlider uses the maximum amount of available space (i.e. the whole player window) for this kind of interaction. In addition and similar to the scale selection, users can not access any random replay speed directly: First, they have to start with a position-based navigation and subsequently, they have to continuously increase replay speed till they reach the actually desired one. In the original work, it is also assumed that the player window uses up only a small fraction of the whole screen thus leaving enough space for horizontal and vertical mouse or pen movements. According to the tests with our implementation of the PVslider interface on a desktop PC, it is questionable if such an interface that is not connected to the window or screen borders can easily be operated on small devices such as PDAs.

3.3 Implementation of the MobileZoomSlider

As already said in the introduction, many of today's mobile devices offer high quality replay of digital video. However, not all of them are powerful enough to support the advanced interactions realized by the MobileZoomSlider. For example, presenting immediate visual feedback from the video while a user is navigating along the timeline requires a lot of processing power but is essential for the proposed video browsing approaches. In addition, not all devices provide a programmable interface that enables developers to do such a low level implementation as it is required in our case.

In addition, when actually implementing the interface design described above, a lot of options exist and decisions have to be made about parameter settings, visualizations etc. All of them are essential for the success (or – if done wrong – failure) of the

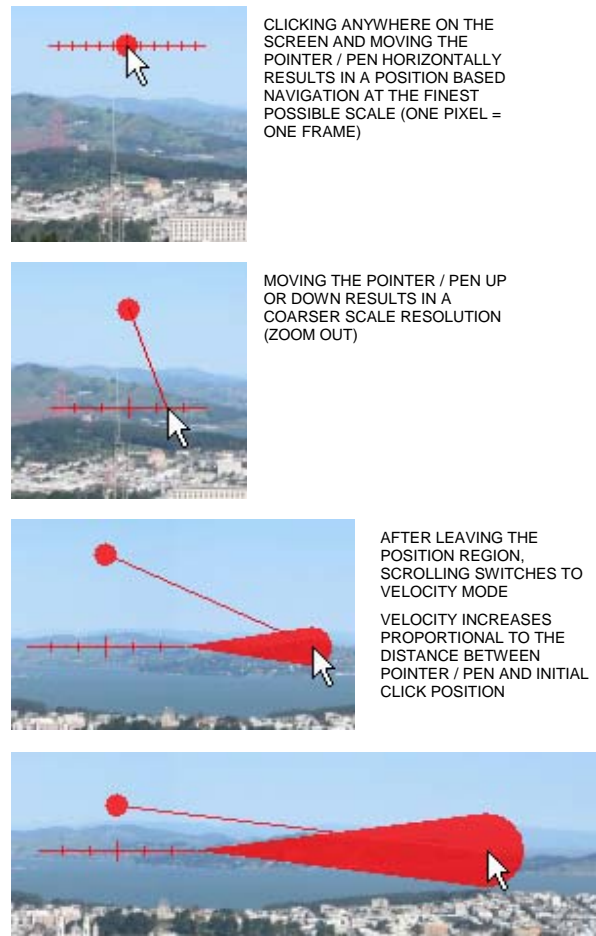


Figure 5. Our re-implementation of the PVslider [17].

interface. In [10, 11] we introduced a slightly different but conceptual similar version of the ZoomSlider for video browsing on desktop PCs. With this implementation, several evaluations have been performed in order to verify its usability and usefulness as well as to optimize important parameter settings. The experience we gained from these evaluations was obviously considered here as well. However, implementation of the MobileZoomSlider differs significantly from the desktop PC version mainly because of the performance issues mentioned above. Other distinguishing features include screen size and pen instead of mouse input (cf. Section 1). The resulting differences and limitations therefore have an impact on both implementation and usability. In the following, we describe the implementation of the MobileZoomSlider in further detail. Usability issues are discussed in Section 4 where we present a first user study.

Based on a detailed analysis of the performance and programming environments for several devices, platforms, and APIs, we decided to use a Dell Axim™ X51v PDA [5] for the implementation of our MobileZoomSlider interface. The Dell Axim™ X51v is a high-performance PDA with an Intel XScale, PXA 270, 624 MHz processor, 64 MB SDRAM, 256 MB Flash-ROM and screen size and resolution of 3.7" and 640x480 pixels, respectively. It also has an Intel 2700g co-processor which



Left: Normal replay mode. *Bottom:* Fullscreen replay mode. In Fullscreen mode, the control units are faded in once the user hits the screen with a pen and faded out after a certain time interval of no user interaction.



Figure 6. Re-implementation of the TCPMP's interface (cf. Fig. 1).

(among other things) supports hardware-side video decoding of several standards. It runs Microsoft's Windows Mobile 5 as operating system.

As player software on top of which to build our interfaces for video browsing, we have chosen "The Core Pocket Media Player" (TCPMP, [4]), a freely available open-source media player which offers high-performance video playback because of its special hardware-optimization for mobile devices. It is available for a variety of platforms, including Windows Mobile and PalmOS.

A snapshot of the standard interface of TCPMP can be found in Figure 1. When we started experimenting with it, we immediately became aware that this interface has most likely not been developed with the advanced interaction and browsing functionality in mind that we are looking for but presumably for a pure passive consumption of video content, since most of the interface elements are rather small and therefore hard to target in an interactive situation. Hence, we started by implementing a re-designed interface with larger widgets for easier interaction. Snapshots of the implemented re-design are shown in Figure 6. In addition to larger control elements, it displays further information which is important for video browsing, such as current replay position (in minutes and seconds) and replay speed.

The position-based navigation of the ZoomSlider is implemented in the same way as described above: If a user clicks at any



Figure 7. Position-based navigation (top) and respective visualization: Icon at the pen tip (left), coarse and fine scrollbars (middle).

position on the video and moves the pen to the left or right, navigation along the timeline is realized with a scale that is proportional to the distance from the lower screen border: Fast scrolling (i.e. coarser scale) at the bottom, slow scrolling (i.e. finer scale) at the top. For the visualization, we have chosen an icon which appears close to the pen anytime a user clicks on the screen and which illustrates the respective functionality (cf. Figure 7). In order to illustrate the different scale resolutions, the scale of the original slider at the bottom of the screen is adapted accordingly: When scrolling at the bottom of the window, the same scale that is originally represented in the timeline of the slider interface is shown. When moving the pen upwards, the ticks representing the scale within the timeline visualize the finer scale that is used at the respective vertical position of the pen (i.e. a "zoom in" into the scale is performed which was also the original motivation to name the interface "ZoomSlider").

The speed-based navigation functionality at the window border is illustrated through a gray icon which appears anytime a user clicks on the screen and which changes its color to white as soon as the pen enters the right area of the player that is reserved for speed-based navigation. Scrolling speed is set according to the horizontal position of the pen as described above. Three different occurrences of the widget represent speed changes as shown in Figure 8. Speed-based scrolling is performed as long as the user holds the pen anywhere in the speed-border. As soon as the pen is released, the player returns to its previous state (i.e. normal replay or pause mode). Figure 9 illustrates the different regions of the overall interface reserved for position- and speed-based navigation, respectively.

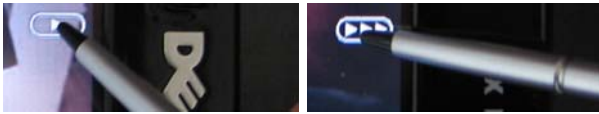


Figure 8. Different icons represent different replay speeds at the right border of the screen (“speed-border”)

Considering performance issues, the implementation of the position-based navigation worked surprisingly well and offers a smooth visual feedback during scrolling even at coarser granularity levels. Regarding the speed-based navigation, there are several limitations, especially for higher replay speeds. Because of buffer underflows for larger speed values, we had to restrict maximum replay to four times normal speed. The smallest replay speed was set to 1/10 times normal replay which is also the established lower limit for traditional VCRs. In addition to the performance problems with higher replay, we realized that there is a slight time-delay caused by the underlying player implementation when moving from the position-based region to the speed-based region. Although this delay does not influence usability, it prevented us from implementing the adapted speed-mapping described above. Instead, we use a linear mapping from the maximum replay speed at the bottom of the player to the minimum replay speed at the screen’s upper border. Another option we originally planned to integrate was backwards scrolling on the left screen border in a similar way as forward scrolling is provided on the right side of the player. Again, this option could not be realized due to performance problems caused by the original player implementation.

4. USER STUDY AND EVALUATION

The MobileZoomSlider offers much more functionality than traditional mobile video interfaces, for example, the one depicted in Figure 1. Hence, it enables users to perform search and browsing tasks which are normally hard to accomplish successfully or can not be solved at all. The critical questions are therefore if users are able to handle this advanced functionality, if the interface can be operated intuitively, and if the interaction experience is successful as well as enjoyable. Differences in the implementation, form factor of the device, pen- instead of mouse input, and performance issues can have a tremendous influence on usability in the mobile case compared to the desktop version. Instead of doing a comparative study with traditional interface designs, we therefore decided to perform a qualitative, informal usability study with the MobileZoomSlider.

Our initial user study included 20 participants (four females, 16 males) from the age of 19 till 38. All of them possess and regularly use mobile devices (mostly cellphones) but only some of them had experience with pen-based handheld devices (mostly PDAs). Since no significant difference between different user groups according to gender, age, or experience with pen-based systems could be observed in the data, these characteristics are ignored in the following discussion.

4.1 Evaluation setup

For the evaluation, the Dell Axim™ and the implementation of the MobileZoomSlider described in the previous section and illustrated in Figure 7 till 9 was used. As video, a commercial that was available for free download from the web was used during the



Figure 9. Regions for different kinds of interaction with the player software: Position-based navigation (top left), speed-based navigation (top right), and standard player widgets (bottom).

whole evaluation. Its length was approximately 64 seconds and it contained 1918 frames (frame rate 29.97 frames per second) and had a resolution of 320x240 pixels. It was shown in fullscreen mode (i.e. rescaled to the PDAs resolution of 640x480 pixels). It had 32 clearly separable scenes, most of which showing a single person talking on the phone. The longest scene was approximately 7 seconds long and contained 210 frames, the shortest one was only about half a second long and contained just 15 frames.

The aim of the evaluation was to figure out, if the users understand and can handle the interface at all, how well they can solve video browsing tasks, and to identify drawbacks, problems, and critical issues as well as to get ideas for further developments and improvements of the MobileZoomSlider.

First, users were given a short introduction into the functionality of the MobileZoomSlider. Then, the device was handed over to them and they were given some time to experiment and play with the system. Based on their first impression, they had to rate the interface in terms of their personal, subjective impression as well as its overall usefulness. In addition, they could give comments, suggest improvements, etc.

Finally, they had to solve two tasks: First, the used video was set to a random position close to the end of the file and the users were asked to reset replay to the very beginning of the file. The purpose of this task was just to verify if they understood the position-based navigation concept as well as the scale-dependency of the horizontal pen position. In order to solve this task in the easiest way one has to start scrolling backwards by placing the pen on the bottom right corner of the screen. In a second task, the participants were asked to count the number of scenes in the video. Since the video contained very short scenes (e.g. half a second long, cf. above), this was a very hard task and we did not expect many users to solve it correctly.

4.2 Results

Users’ ratings. Since a lot of video clips, especially on mobile devices, are used for infotainment purposes (i.e. for entertainment as well as to gather information), it is very important to measure not only the usefulness of an interface but also the personal impression of the users. Hence, we asked them two questions in order to rate the interface. Almost all users gave a rather high

ranking for the first question, i.e. “How would you personally rate this interface on a scale from 0 (= disapproval) to 3 (= pleasant and enjoyable)?” (average rating: 2.35). Detailed results are given in Figure 10. Figure 11 shows the results to the second question, i.e. “How would you rate the usefulness and added value of this interface on a scale from 0 (= not useful) to 3 (= very useful)?” The average rating for this question was 2.1. Only one user gave a rather low rating (i.e. “1”) in both cases. However, this was a rather critical user who gave very useful comments. His main reasons for the low rating have been the visualization of the scale within the timeline of the original slider and the orientation of the scale resolution for position-based navigation. We will comment on these two issues below. Six users liked the interface, but gave a little lower rating for its usefulness, whereas three rated the usefulness a little higher (difference in the rating of +/-1 value). Only one user gave the highest rating for the first question but rated the usefulness with “1”. Although ratings for the usefulness are a little lower in general, we were still very surprised about the very positive feedback. No user gave the lowest possible rating (“0”) for any of the two questions.

Intuitiveness and handling (Task 1). Generally, we can say that all users were able to handle the interface easily although they had no significant training time. A few participants needed a little more time to make themselves familiar with all the functionalities offered by the system, but all were able to handle it very well. (Some explicitly mentioned that it takes some time to get used to and requires some training but they all agreed that it is intuitive and not too complex to handle after you experiment with it for a while.) Only in one case it was necessary to give some hints on how to solve the first task, i.e. reset replay to the beginning of the file, in the easiest and quickest possible way. All other users were able to solve it in a reasonable way without any further instructions.

Actual usage (Scene counting, task 2). As said above, we have purposely chosen a task that was hard to solve correctly, so we did not expect many users to find the correct answer (i.e. 32 scenes). During the evaluation we also had the impression that some users did not consider the possibility that a scene can be as short as half a second and therefore, most of the answers were a little below the correct value (average 26.05 scenes). Detailed results are shown in Figure 12. Except for two users who only identified 10 and 13 scenes, respectively, most of the answers were quite reasonable given the complexity of the task. It is hard to judge if these two were just too impatient (or unmotivated) to solve this task correctly or if they really had problems with the operation of the interface. Our subjective impression was that they did not want to spend too much time on this task and therefore did not put enough effort into its solution. Both of their ratings confirmed to the average ratings of the other users except for one of them who rated the usefulness rather low (i.e. with “1”). Nevertheless, the results illustrate the general suitability of this interface for such a browsing task. Another very interesting observation is that all but two users explicitly used position-based navigation to solve this task although a speed-based navigation strategy would also be very appropriate here. We will comment on this issue in the next subsection.

4.3 Users’ Comments and Discussion

From the ratings, the results of the two tasks the users had to solve, as well as based on their personal comments during and after the evaluation, we can conclude that the proposed interface

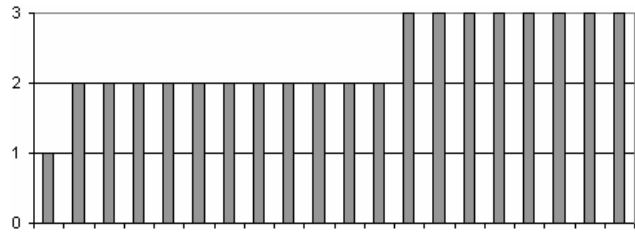


Figure 10. Rating of personal, subjective impression (users vs. rating from 0 to 3).

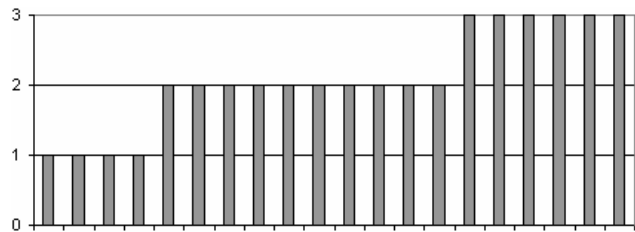


Figure 11. Rating of the overall usefulness of the interface (users vs. rating from 0 to 3).

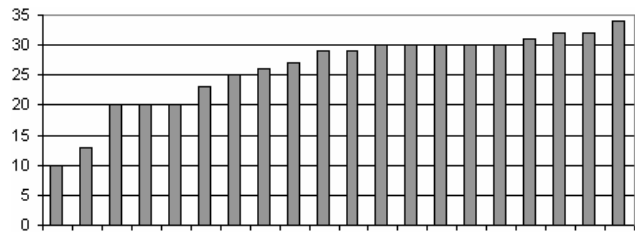


Figure 12. Results for task 2 (users vs. number of detected scenes).

for video browsing on a pen-based mobile device is intuitive, easy to learn and operate, and enables users to quickly solve several browsing tasks. However, there is obviously still room for improvement and many users made good comments that pinpointed open problems and disadvantages of the current implementation and highlighted issues for further developments. In the following, we summarize the most important observations.

Position-based navigation and scaling. As said above, one reason why one user gave a rather low rating was that he felt the orientation of the scale resolution (i.e. fine scale at the top and coarse scale at the bottom) was not intuitive. Instead, he would have expected it the other way around. Four other users mentioned this issue as well although it did not influence their final rating very much. When asked why, no one could give us a reasonable answer. Hence, we assume that it is just a personal preference or bias, and the problem can probably be solved by making this a parameter that can be configured by the users themselves. (Our motivation for the used orientation was that this way the coarse scale is close to the timeline of the original slider which has the same resolution.)

Another issue in this context is the granularity of the scale in between the maximum and minimum resolution at the bottom and top of the screen, respectively. In the current implementation, we used a linear interpolation between these two values. During the

evaluation we had the impression, that it might actually be better to offer only a few discrete values instead of a continuous range of different scales. The reason for this is that (in the same way as the resolution of the horizontally mounted timeline of the original slider is limited) not all possible scale values between the maximum and minimum value can be mapped onto the screen due to its small size. Hence, we assume that it might be better to just provide a medium scale and one or two further values in addition to the maximum and minimum resolution. One user actually mentioned a similar idea during the evaluation.

Speed-based navigation. Some users noted that it might be better to be able to change replay speed constantly, i.e. to activate and explicitly deactivate it, instead of having to continuously hold the pen down. In addition and similarly to the position-based navigation, it might be worth thinking about offering only a few, discrete speed values instead of a continuous speed range between the maximum and minimum speed, although no user commented on this issue. Generally, it was very interesting for us to observe that all except for two users did not use speed-based navigation at all when solving the second task. Some users even mentioned that if they have the option to do a position-based navigation they would hardly use speed-based navigation at all. We were not able to identify any issue in the implementation leading to this strong preference neither did any of the users make any comments in that direction. In Section 2, we already argued why position-based navigation is important and in some situations even superior to speed-based browsing. Nevertheless, we were quite surprised about this strong preference of the users and their clear comments on this issue.

Visualization. Although the visualization might seem like a rather unimportant issue that is more influenced by esthetics than scientific guidelines, we should not underestimate its relevance for the success (or failure) of an interface. And in fact, the visualization of the actual scale resolution was the second main reason for the low ratings of one user to both questions. He argued, that a finer scale should have more ticks than a coarser scale (because you can actually access more frames). In the implemented version, the complete opposite is the case, because if you always show the same ticks, than obviously the distance between the ticks gets larger if you zoom into the scale and hence, fewer ticks appear in the segment of the scale that is represented on the screen. None of the other participants mentioned this issue. However, we had the impression that only few users really understood the intention behind the visualization chosen by us. Hence, for the future we should come up with a better solution to illustrate the actual scale resolution when a user moves the pen within the position-based region of the screen. One user mentioned that one should consider displaying some information about the scale resolution directly at the pen tip. Others, however, argued that it is better not to represent more information on the screen because it would be disrupting and interfere too much with the content of the video. One user proposed to use transparent widgets for everything that is presented on the screen. We initially tried that but discarded the idea because of difficulties in the implementation.

Further comments and observations. Despite all the critical comments that highlighted problems and gave ideas for future developments, we also got lots of positive feedback. For example, two users expressed their liking of this interface because they can click anywhere on the screen when they want to navigate instead

of having to explicitly target a very small, moving target (i.e. the slider thumb). Although few users used the speed-based navigation in the actual task, we realized that some users actually appreciated its placement at the right window border: Because the screen is mounted a little lower than the actual body of the device, there is a physical resistance at the window border against which the pen can easily be pressed while moving it up and down when manipulating replay speed making it very easy and comfortable to move your pen up and down. Some of the participants came up with their own ideas for situations in which our interface could be useful. For example, one user said “This would be great on a cellphone, so I can shoot a video and then extract single frames from it and send them to my friends.”

5. CONCLUSION AND FUTURE WORK

In this paper we presented the MobileZoomSlider, a new interface for video browsing on pen-based handheld devices such as PDAs and some mobile phones. We discussed our work in relation to existing approaches and presented a first user study that proved the concept and usefulness of our design and also indicated reasonable steps for future work. The main advantage of our approach is that it provides a large range of functionality – position-based navigation at different granularity levels and speed-based browsing at different replay speeds – without overloading the interface and using too much space for widgets on the screen. Although not many participants in the study used speed-based browsing for the particular task they had to solve (and some expressed themselves rather negative about this kind of interaction) we believe that there are lots of situations where this kind of interaction is useful and therefore its seamless integration into the overall interface design can be seen as another important advantage.

In addition to the positive feedback, the participants of the study also gave valuable hints for improvements and further developments, most importantly considering the scale resolution, its orientation, and the visualization, all of which we are currently addressing in a re-implementation of the interface. In addition, we are working on a similar version for the Palm Treo™ Smartphone [15]. Once both implementations are finished, we will perform another user study with both devices.

Further directions for future work include integration of audio feedback during speed-based browsing for replay speeds between 0.5 and 2.5 times normal replay speed. In earlier evaluations with video and audio browsing [9], we identified that such a feedback can be quite helpful in some search tasks if signal processing is used (e.g. [19]) in order to avoid high pitches during faster replay. Another issue which we believe is very useful for navigation and browsing (but to our surprise was not mentioned by the participants of the study) is scene-based navigation. Enabling users to quickly jump from one scene to another or the beginning of the current scene is a very useful functionality for video browsing. One idea to integrate this feature into the interface (without the need to present additional buttons on the screen) is to split the screen in two halves and interpret a pen tip (in contrast to a tip and drag, which evokes the position-based navigation) on the left and right side of the screen as a jump to the previous or next scene, respectively.

6. REFERENCES

- [1] C. Ahlberg and B. Shneiderman. The Alphaslider: A compact and rapid selector. *Proceedings of the SIGCHI conference on Human factors in computing systems*, ACM Press, 1994, pp. 365-371.
- [2] Apple – iPhone: see <http://www.apple.com/iphone/>
- [3] Y. Ayatsuka, J. Rekimoto, and S. Matsuoka. Popup Vernier: A tool for sub-pixel-pitch dragging with smooth mode transition. *Proceedings of the 11th annual ACM symposium on User interface software and technology*, ACM Press, 1998, pp. 39-48.
- [4] The Core Pocket Media Player: see <http://tcpmp.corecodec.org>
- [5] Dell Handhelds, see <http://www1.euro.dell.com/content/products/category.aspx/handheld>
- [6] A. Girgensohn, J. Boreczky, and L. Wilcox: Keyframe-Based User Interfaces for Digital Video. *IEEE Computer*, Vol. 34(9), pp. 61-67, 2001.
- [7] C. Gurrin, L. Brenna, D. Zagorodnov, H. Lee, A.F. Smeaton, and D. Johansen: Supporting Mobile Access to Digital Video Archives without User Queries. *Proceedings of MobileHCI 2006*, September 12-15, 2006, Helsinki, Finland, pp. 165-168.
- [8] W. Hürst, G. Götz, T. Lauer: New methods for visual information seeking through video browsing. *Proceedings of the 8th International Conference on Information Visualisation*, IV 2004, IEEE Computer Society, pp. 450-455.
- [9] W. Hürst, T. Lauer, C. Bürfent, and G. Götz: Forward and Backward Speech Skimming with the Elastic Audio Slider. *Proceedings of the 19th British HCI Group Annual Conference*, Edinburgh, Scotland, UK, 2005.
- [10] W. Hürst and P. Jarvers: Interactive, Dynamic Video Browsing with the ZoomSlider Interface. *Proceedings of the IEEE International Conference on Multimedia & Expo*, Amsterdam, The Netherlands, 2005.
- [11] W. Hürst: Interactive audio-visual video browsing. *Proceedings of the 14th Annual ACM International Conference on Multimedia*, Santa Barbara, CA, USA, 2006.
- [12] H. Liu, X. Xie, W.-Y. Ma, and H.-J. Zhang: Automatic Browsing of Large Pictures on Mobile Devices. *Proceedings of ACM Multimedia 2004*, New York, NY, pp. 148-155.
- [13] T. Masui: LensBar – Visualization for browsing and filtering large lists of data. *Proceedings of the 1998 IEEE Symposium on Information Visualization*, INFOVIS 1998, IEEE Computer Society, pp. 113-120, 1998.
- [14] K. O'Hara, A.S. Mitchell, A. Vorbau: Consuming video on mobile devices. *Proceedings of the SIGCHI conference on Human factors in computing systems*, ACM Press, 2007, pp. 857-866.
- [15] Palm Treo™: see <http://www.palm.com/us/products/smartphones/>
- [16] R. Pea, M. Mills, J. Rosen, K. Dauber, W. Effelsberg, and E. Hoffert: The Diver Project: Interactive Digital Video Repurposing. *IEEE MultiMedia*, Vol. 11, Issue 1, Jan-March 2004, pp. 54-61.
- [17] G. Ramos and R. Balakrishnan. Fluid interaction techniques for the control and annotation of digital video. *Proceedings of ACM UIST 2003*, Vancouver, BC, Canada, Nov. 2003.
- [18] H. Richter, J. Brotherton, G.D. Abowd, and K. Truong: A Multi-Scale Timeline Slider for Stream Visualization and Control, *GVU Center, Georgia Institute of Technology, Technical Report GIT-GVU-99-30*. June 1999
- [19] S. Roucus, A. Wilgus: High quality time-scale modification for speech. *IEEE International Conference on Acoustics, Speech, and Signal Processing*, Vol. 2, 1985.
- [20] S. Srinivasan, D. Ponceleon, A. Amir, and D. Petkovic: "What is in that video anyway?": In Search of Better Browsing. *Proceedings of the IEEE International Conference on Multimedia Computing and Systems*, Vol. I, 7-11 June, 1999, Florence, Italy.
- [21] Y. van Houten, E. Oltmans, and M. van Setten: *Video Browsing and Summarization*. Enschede, Telematica Insitute, January 2001. Available at <https://extranet.telin.nl/docuserver/dscgi/ds.py/ViewProps/File-12409>
- [22] 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>
- [23] H.D. Wactlar, T. Kanade, M.A. Smith, and S.M. Stevens: Intelligent access to digital video: Informedia project. *IEEE Computer*, 29(5), 46-52, 1996.