

Wolfgang Hürst  
Utrecht University

## Video Browsing on Handheld Devices—Interface Designs for the Next Generation of Mobile Video Players

Recent studies confirm that mobile video use on handheld devices is quite different from watching TV in your living room.<sup>1</sup> For instance, assume you are standing at the bus stop, waiting for the 10-minute bus ride that will take you home from work. While waiting, you want to quickly view the latest evening news show, which you downloaded to your mobile device just before you left your office. Once you're on the bus, you want to quickly go to the one news report that seems the most interesting to you, because you don't have enough time to watch the whole news show. Shortly before you arrive, you want to take a quick look at the weather forecast. Because you don't have much time left, you just want to find a frame in the video showing the map of your state with the temperature values. When comparing such a situation to watching this news show at home, we can see that in the mobile scenario, much more interaction and navigation occurs. First, you have to quickly skim the video content to get an overview. Then you have to set replay around the beginning of the particular news clip you want to watch. Finally, you need detailed navigation—that is, the ability to move to an exact position or a particular frame within the video.

The need to continuously switch between different granularity levels when navigating through a video (skimming on a coarser level, for example, to get a quick overview versus detailed browsing on a fine level to do an exact positioning) creates high demands on interface design and functionality. Researchers have conducted many studies on video browsing, including, for example, looking at automatic generation of trailers and storyboards; good overviews are available elsewhere.<sup>2-6</sup>

However, most of these techniques take place on desktop machines and can hardly be applied to mobile devices because of limited screen size and no mouse-like input device. Surely, the keypads and joystick-like buttons of modern mobile phones can be used for video browsing as well, but they hardly can provide the flexibility needed to cover the whole range of interactions that must be performed for complex browsing tasks, such as those required in the scenario described above. A better approach seems to use a finger or a pen to interact with the (touch sensitive) screen directly, allowing us to create software-based interfaces that can be adapted to the actual demand of a particular program. An impressive example of this strategy is Apple's iPhone (and the iPod touch, which features a similar interface). Innovative interface concepts, such as multitouch and flicking (see "Static Media Browsing on the iPhone and the iPod Touch" sidebar near the end of this article) are indeed one of the main reasons for the hype generated around the iPhone. However, this functionality is normally applied to browsing of static media, such as lists, text, and images, whereas the included video player offers limited possibilities for interactive navigation, at least at the time of this writing (see Figure 1).

This article gives an overview of our group's recent and ongoing work on creating a flexible, intuitive, and powerful interface to improve video browsing on handheld devices in a similar way to how the iPhone's interaction techniques revolutionized navigation in mobile static media. I present several concepts and related user-interface designs that we've developed and evaluated. Due to the overview

character of this article, I limit the discussion to the presentation of the designs and refer to the related publications for detailed descriptions of evaluations and user studies.

We implemented the interfaces presented here on a Dell Axim X51v PDA running Windows Mobile. The PDA features an Intel 624-MHz XScale PXA 270 processor, 64 Mbytes of SDRAM, and a 3.7-inch, 16-bit, 640 × 480 display. We implemented the designs on top of The Core Pocket Media Player (TCPMP), an open-source media player available for different platforms, including Windows Mobile and Palm OS. Implementation was done in C++ and based on the Win32 API using the Graphics Device Interface for rendering. The Dell Axim PDA provides a touch-sensitive display that is optimized for pen-based input. Hence, all interfaces have been designed for and tested with pen-based interaction. Some of the designs can be generalized to finger-based input whereas others are most likely unsuited for this kind of interaction due to a finger's larger size compared to a pen's fine tip.

### MobileZoomSlider design

Some traditional video browsing techniques—such as automatic trailer generation, story segmentation, and so on—are also useful approaches for mobile devices, for example, to get a quick overview of a video's content or to skip a scene of minor interest. However, as with hardware-based solutions such as buttons and joystick-like keys, these techniques aren't sufficient for providing the whole range of functionality used for the advanced browsing tasks illustrated in the example at the beginning of this article. Approaches such as interactive manipulation of replay speed (for example, fast forward and slow motion to skim a file's content quickly and do an exact positioning on frame level) or a flexible navigation along the timeline as realized with a timeline-based slider (see Figure 1) seem to be more appropriate. The latter one has proven to be a useful approach for video browsing when combined with real-time visual feedback while a user is dragging the slider's thumb along the timeline.

Getting immediate feedback enables users to skim a file at random speed and direction and thus allows them, for example, to browse a file's content quickly (for example, to get a quick overview of all news messages), easily set



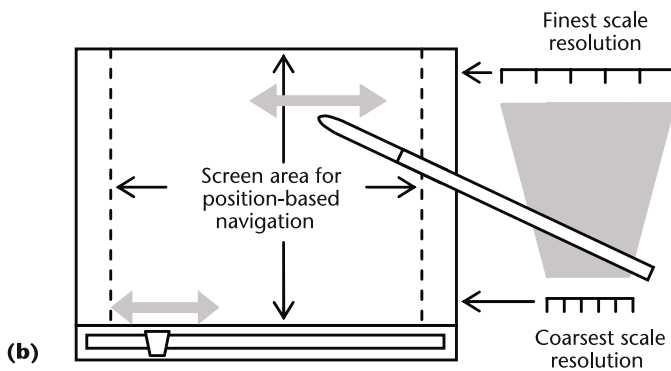
replay to a specific area (for example, the approximate beginning of the most interesting news story), and do an exact positioning (for example, to access one of the few frames in the video showing the map with the temperatures for the next day). Such a slider-based, video-browsing approach would therefore be useful for typical mobile video-browsing tasks. However, the small screen size of handheld devices prevents us from using it very easily. First, small icons often are hard to target even with the fine tip of a thin pen. Second, and most importantly, sliders don't scale to large document sizes. Moving the slider's thumb for the smallest possible unit on the screen (that is, one pixel) already results in a jump of several frames in the video if the document is rather long, making it difficult if not impossible to do a fine granular navigation.

The first interface design we implemented, the MobileZoomSlider (see Figure 2a, next page), addresses this scaling problem by providing different sliders with several scales at various granularity levels. In addition, it eliminates the need to target small icons. Instead of relying on particular widgets or GUI components, users can click anywhere on the display. Navigation along the timeline is evoked by moving the pen horizontally, as Figure 2b illustrates. The scale resolution of the corresponding virtual timeline depends on the pen's vertical position: the finest resolution (one pixel on the screen corresponds to one frame in the video) is achieved at the top

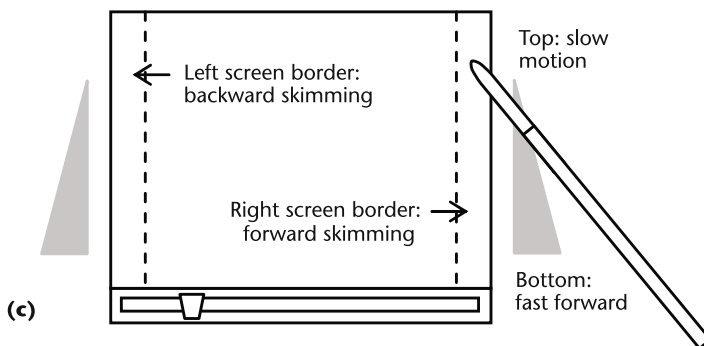
*Figure 1. The iPod touch's video player (iPod touch ver. 1.1.4). Users can skip scenes using two buttons and navigate through the file along the timeline using a slider at the top of the window. Browsing granularity is high due to the small size of the screen, thus preventing users from performing detailed browsing and exact positioning.*



(a)



(b)



(c)

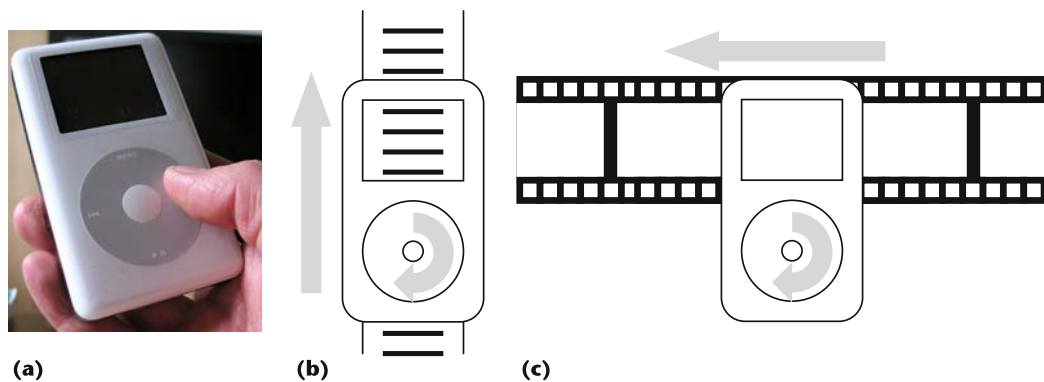
**Figure 2.** The MobileZoomSlider interface design: (a) implementation on a Dell Axim PDA, (b) position-based navigation in the screen's center, and (c) speed-based navigation at the screen's borders.

of the display. The coarsest resolution (one pixel on the screen corresponds to the number of frames in the video divided by the horizontal resolution of the screen) is associated with the bottom of the screen. In between, the timeline's scale is linearly mapped to a value between these two extremes. This way, a user can access the timeline easily to scroll through the video at different granularity levels and immediately switch between different scales.

When reaching the right-screen border, browsing changes from active navigation along the timeline to passive video browsing by manipulation of replay speed. As with the scale resolution when browsing along the timeline, the actual speed value at the screen's border depends on the vertical pen position: slow motion for fine granular navigation at the top versus fast forward for quick document skimming at the bottom, and a linear interpolation of replay values in between (see Figure 2c). Backward replay at different speed values is similarly implemented on the left-screen border.

The MobileZoomSlider was initially introduced in Hürst, Götz, and Welte<sup>7</sup> as a slightly modified version of our original ZoomSlider interface, which was developed for video browsing on desktop PCs and laptops.<sup>8</sup> The related usability study illustrated the power and flexibility of the proposed approach and verified its usability for typical browsing tasks. The functionality offered by this design is a combination of two traditional interaction concepts for video browsing: position-based navigation (that is, modification of the current position in a file by horizontal pen movements along the virtual timelines) and speed-based navigation (that is, modification of replay speed by vertical pen movements at the left and right border of the screen). Hence, the innovation in this interface comes less from the interaction concepts than from their clever integration into the overall design. Users don't have to target any small icon to evoke some functionality; instead, they can just click at the respective area of the screen. For position-based navigation, interaction is similar to traditional slider movements: left and right pen movements for backward and forward navigation along the timeline.

The timeline's granularity is clearly associated with fixed screen areas, giving users immediate access to the appropriate scale resolution needed in a particular browsing situation. The areas on the left and right side of the screen, which are reserved for speed-based browsing, resemble a normal (horizontal) slider sometimes used in video editing or replay tools on desktop PCs. However, here, the functionality is evoked by vertical pen movements. The reason for this is to enable users to switch easily between the two different interaction modes: speed- and position-



*Figure 3. (a) The iPod classic's Click Wheel. Positions in the file or frames along the timeline are mapped onto positions on the circular wheel, enabling users to scroll forward and backward through (b) files or (c) videos by turning the wheel clockwise and counterclockwise.*

based navigation. The involved parameters—scale resolution of the virtual timeline and speed-up factor—are harmonized with each other. A slow movement along a fine granular timeline at the top of the screen leads to slow-motion-like, speed-based navigation when moving the pen to the display's right side (and vice versa). By the same token, navigation along the timeline at the bottom of the screen (which is usually faster due to the lower resolution of the timeline) is associated with a faster replay at the bottom right corner of the screen. Our evaluations showed that users often take advantage of both interaction modes when confronted with advanced browsing and search tasks, thus confirming the need for their smooth integration into the overall interface design.<sup>9</sup>

### ScrollWheel design

Allocating different functionalities to certain areas of the screen as with the MobileZoomSlider is not a new idea and is also used, for example, in the iPhone for browsing long lists of text (see Figure A3 in the sidebar). Our second interface design for mobile video browsing, the ScrollWheel,<sup>9</sup> also has an interesting relation to an Apple device, in this case the iPod classic, which features the Click Wheel. This interface element is used not only to control volume, but also to navigate static media. And if we linearly map each frame in a video to a position on the wheel (corresponding, for example, to the hour markers listed on an analog clock) we also can use such a device for video browsing (see Figure 3). However, in our case, we want a software version of such a wheel, which has an additional advantage compared to a hardware installation. By modifying the circle's radius, users can indirectly manipulate the resolution of the associated scale so that larger circles result in slower

movements along the timeline and smaller circles can be used for quick navigation with a coarser timeline resolution. This behavior is comparable to the MobileZoomSlider design where users navigate along the timeline in one case via linear pen movements and in the other via circular motions. The larger the distance between the pen and the screen's bottom or center of the circle, respectively, the finer the scale of the timeline.

In an initial experiment, we compared different versions of such a ScrollWheel implementation on our Dell Axim PDA:<sup>9</sup>

- a pure position-based version (see Figure 4a, next page) where the wheel resembles a virtual clock with a resolution that can be manipulated indirectly by increasing and decreasing the distance between the pen and the center of the wheel,
- a pure speed-based version (see Figure 4b) where turning the wheel clockwise and counterclockwise results in an increasing scrolling speed in forward and backward directions, respectively, and
- a combined version (see Figure 4c) where interaction starts with a position-based navigation and after a while switches to speed based scrolling.

The latter one is comparable to the change of interaction modes in the MobileZoomSlider design once the pen approaches the left or right side of the screen. However, in a heuristic evaluation of all three designs, we discovered that it might be too complex to handle; it seems better to separate both interaction modes instead of combining them in a single interface element.

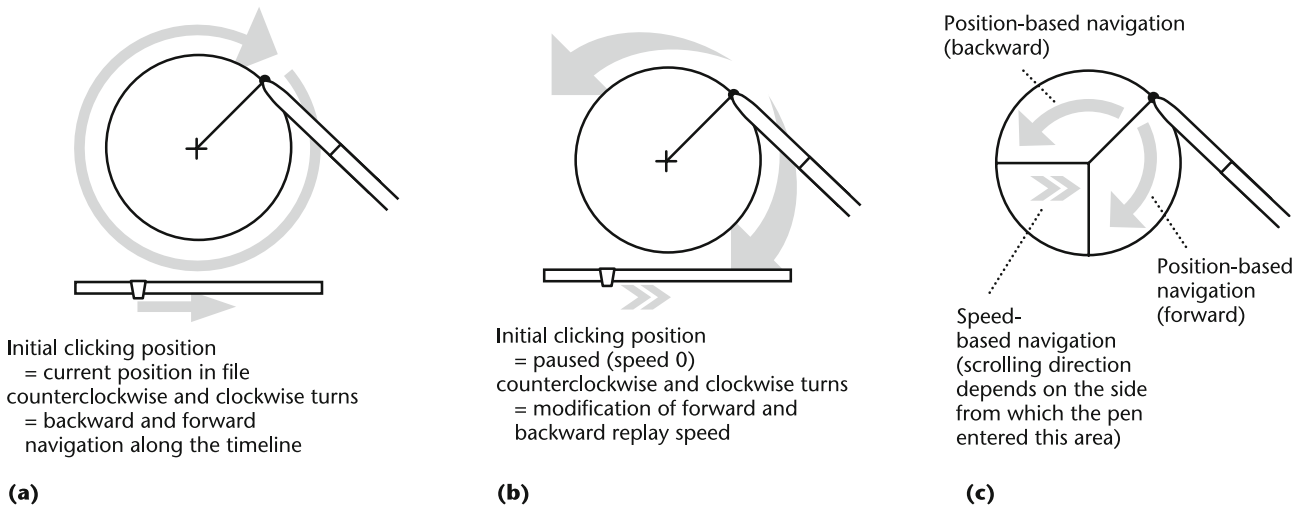


Figure 4. Different versions of the ScrollWheel implementation: (a) position-based navigation, (b) speed-based navigation, and (c) combined version.

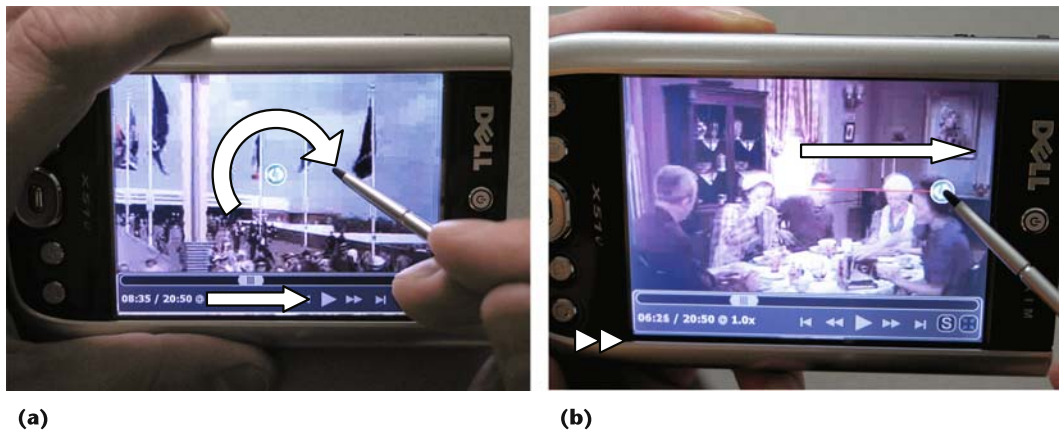
On the basis of the results of the initial evaluation, we implemented a revised version of the ScrollWheel where the wheel itself is used purely for position-based navigation, as it was in the first version from the previous list. Users achieve speed-based navigation by horizontal pen movements; clicking into the wheel's center and moving the pen to the left or right results in an increase of replay speed in backward and forward direction (see Figure 5). We evaluated this implementation in a comparative study with the MobileZoomSlider design. The experiment confirmed that the strict separation of interaction styles (circular and horizontal pen movements for position- and speed-based navigation) improves ScrollWheel usability. In addition, we found initial evidence that users might prefer the circular movements for position-based navigation over the horizontal movements required by the MobileZoomSlider, whereas the MobileZoomSlider's speed-based navigation was sometimes preferred over the version implemented in the

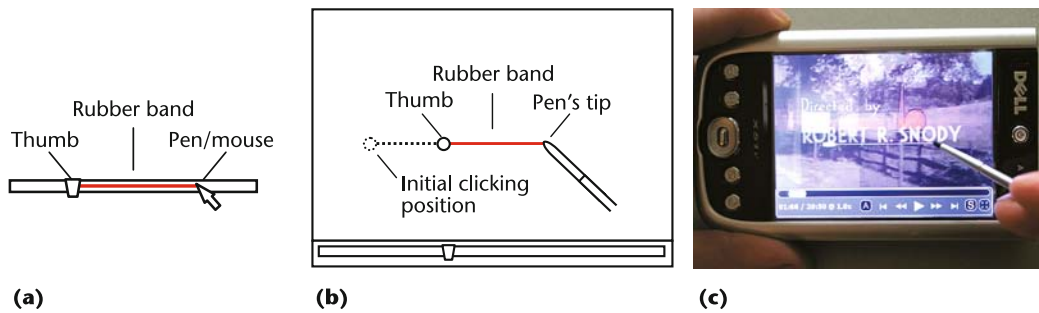
revised ScrollWheel. Hence, a combination of the ScrollWheel's position-based browsing mode with the MobileZoomSlider's speed-based navigation seems to be the interface of choice for most participants of our experiment. However, further evaluations are needed before we can generalize these observations.

### ElasticSlider design versus iPhone-like flicking

In our latest implementation,<sup>10</sup> we compared the so called ElasticSlider design with an approach for video browsing that resembles the flicking used to scroll through large lists of text on the iPhone or the iPod touch (see the sidebar). Elastic sliders were originally introduced for navigation in static documents, such as lengthy text or lists with a large number of entries.<sup>11</sup> We adapted them for video browsing on desktop PCs.<sup>12</sup> The basic idea is that users don't grab the thumb of a slider directly, but instead pull it along the timeline via a virtual rubber band that spans the space between the

Figure 5. Implementation of the revised version of the ScrollWheel: (a) position-based navigation and (b) speed-based navigation.





*Figure 6. Illustration of the ElasticSlider interface design: (a) basic idea of the elastic slider (scrolling speed is proportional to the rubber band's length (the longer the band, the stronger the tension, and hence, the faster the scrolling), (b) on-screen version, and (c) implementation on a PDA.*

thumb and the mouse pointer. The thumb follows the pointer with a speed proportional to the rubber band's length—that is, a larger distance increases the tension on the rubber band and thus scrolling speed, whereas a smaller distance decreases the tension on the rubber band resulting in a slower movement of the slider's thumb (see Figure 6). As with the MobileZoomSlider, this elastic scrolling functionality is not coupled to a particular GUI in our implementation; rather, it is evoked by clicking directly on the screen and moving the pen to the left or right. The implementation on our Dell Axim PDA is similar to the one described in Hürst, Götz, and Lauer<sup>11</sup> with small differences, for example, in the mapping of distance-to-scrolling speed and the visualization: Because of the small screen size, we decided to keep the illustration of the interface on the screen to a minimum to avoid blocking significant parts of the video during browsing.

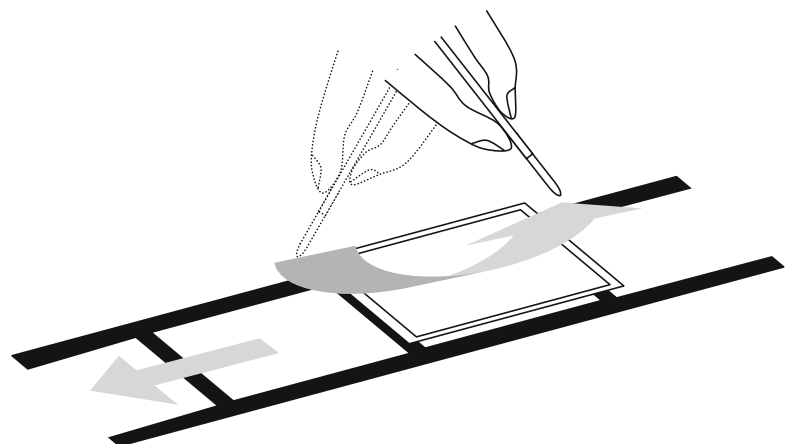
As with the ElasticSlider, iPhone-style flicking can be explained on the basis of a physical movement characteristic: flicking the pen over the screen (or the finger in case of the iPhone) pushes the content of the file's visualized part (the current frame of a video or the visible parts of a text) in the same direction as the pen is moved. Scrolling speed depends on the momentum of the pen flick (scrolling gets faster the harder the user pushes the pen). It decreases after a while, simulating a frictional loss effect. For video browsing on our PDA, we implemented a slightly modified version of the iPhone-style flicking where the user doesn't push the actual video and instead scrolls the file by flicking the pen in the direction a related slider thumb would move along the timeline. That is, scrolling and pushing direction behaved reversely compared to the original iPhone implementation because for video, such a realization seemed more intuitive to us (see Figure 7).

In the comparative study of these two interface designs, users had to solve different browsing problems that were quite similar to the three tasks described in the introductory example at the beginning of this article. We measured the time participants took to solve these tasks as an objective evaluation criterion and recorded subjective user opinions and some ratings given for different aspects of the interfaces. Interestingly, there was only a minimum difference in the average ratings given to both designs. However, the distribution of the single ratings differed significantly. Whereas the ratings for the ElasticSlider were almost all quite close to the average value, iPhone-style scrolling obviously polarized the users into two groups: one was strongly in favor of this kind of interaction and the other gave it a lower rating compared to the ElasticSlider. It was quite surprising to us that in the actual performance test, both interfaces performed equally well; we observed no significant difference.

### Summary and outlook

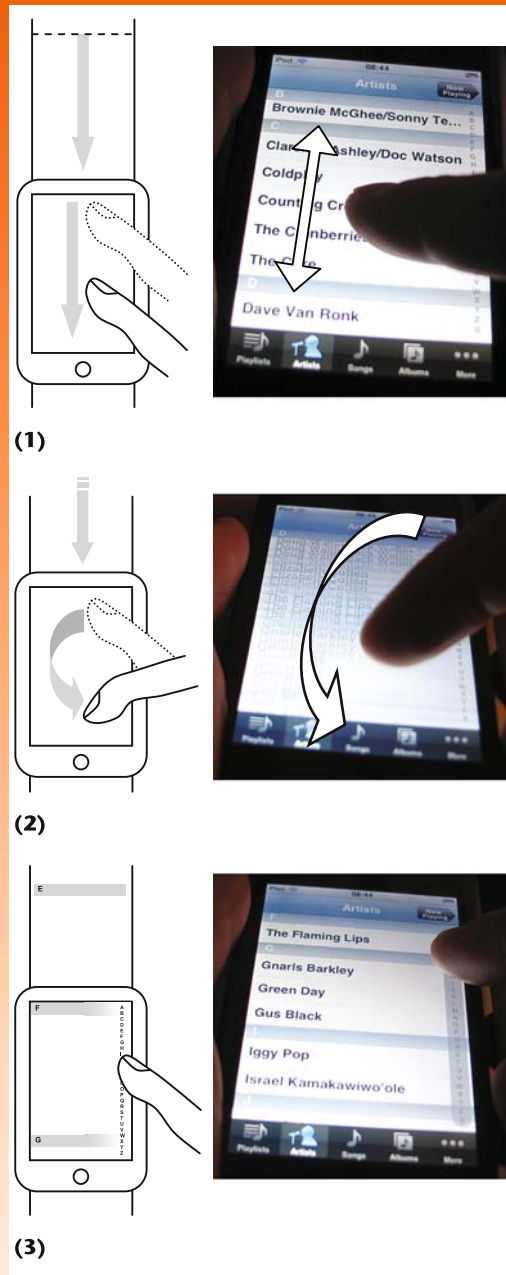
Recent and ongoing work on better interfaces for video browsing on handheld devices is essential for increasing the usability of

*Figure 7. Illustration of the iPhone-style flicking for video browsing. By flicking the pen over the screen, users can navigate forward and backward through a video. Initial scrolling speed and the decreasing factor depend on the momentum of the pen's flick. Our implementation calculates this momentum according to the speed at which the user moves the pen over the screen.*



## Static Media Browsing on the iPhone and iPod Touch

Figure A. (1) By holding and moving the finger on the screen, users can manipulate the visible part of the file by moving it up and down. The maximum distance to scroll is one height of the screen, as illustrated by the dashed line. (2) By flicking the finger over the screen, users can scroll through a long list or text document at different speeds (granularity levels). (3) By touching the screen's right border, users can access an alphabetic index enabling them to jump directly to the corresponding part of the list.



One of the most significant contributions of the iPhone and the related iPod touch is the introduction of some innovative interface concepts. For example, users can zoom in and out by tapping or using multitouch input, thus enabling them to interactively adapt the visible content to the small screen. Whereas these techniques for zooming are mainly intended for manipulation of static content (images and text), some of these innovative approaches for scrolling could be useful for video browsing as well.

On a desktop, long lists of text, such as thousands of titles from a private music collection, are normally browsed using a scrollbar. For small mobile devices, such an approach is normally unsuitable. The small screen size not only makes it difficult to target and operate such a tiny interface element, but also results in a resolution of the scrollbar's scale that is too coarse to access each single list entry directly. Not all elements from the list can be mapped to an individual position on the scrollbar. The iPhone and iPod touch deal with this problem by introducing a combination of browsing techniques. Placing the finger on the screen and moving it up and down simulates moving the file's content behind the visible area, as illustrated in Figure A1.

However, if the user flicks the finger, that is, touches the screen and quickly pushes the file upwards or downwards before releasing it, the file keeps scrolling, as Figure A2 illustrates. Initial scrolling speed depends on the momentum of the finger's flick and then slowly decreases. By modifying this momentum, users can scroll through a long list of text at different speeds. One problem with this approach is that even if you flick a

file's content very quickly to cover larger distances, getting to the end of a long list might require many finger flicks. Figure A3 illustrates one way to deal with this problem: if the user touches the screen's right border, an index appears, allowing the user to access particular parts of the alphabetically sorted list by selecting the respective letter. Such an index resembles a coarse scrollbar.

The combination of these three interaction techniques provides users with a wide range of scrolling functionality at different granularity levels. They can do an exact positioning on a very fine scale by placing the finger on the screen and moving it up or down (see Figure A1), skim through a list at different speeds by flicking the finger over the screen with different momentums (see Figure A2), and do a rather coarse navigation by skipping larger parts of the list with the letter-based index on the screen's right side (see Figure A3).

mobile video. Our latest evaluation—which compared the ElasticSlider design with iPhone-like flicking—illustrates two important issues in this context. First, subjective assessment plays an important role and differs among users. Second, neither of the two interfaces could clearly outperform the other in the objective evaluation. Although we have not yet performed any comparative study among all interface designs discussed in this article, we are pretty sure that such an evaluation would also not identify a clear winner.

Whereas the laboratory studies done with the proposed designs clearly illustrated their usability and demonstrated their intuitiveness and powerful functionality, several long-term studies conducted in real-world conditions are needed to identify the best possible solution and better evaluate the advantages and disadvantages of the different implementations. In addition to working on such projects, we plan to include the integration of further video browsing approaches—such as scene-based navigation—into the proposed designs. Also, we are investigating the applicability of the presented interfaces for finger-based interaction. For example, the MobileZoomSlider design doesn't seem to be very useful for such situations because the finger blocks too much of the video content. Initial tests with the other designs seemed quite promising. **MM**

## Acknowledgments

The author thanks Georg Götz and Konrad Meier from the Albert-Ludwigs University, Freiburg, Germany, who participated in the development, implementation, and evaluation of the described interfaces. The video used in the images of this article was taken from the Prelinger Archives and is available as public domain from the Internet Archive at [http://www.archive.org/details/middleton\\_family\\_worlds\\_fair\\_1939](http://www.archive.org/details/middleton_family_worlds_fair_1939).

## References

1. K. O'Hara, A.S. Mitchell, and A. Vorbau, "Consuming Video on Mobile Devices," *Proc. SIGCHI*

- Conf. Human Factors in Computing Systems*, ACM Press, 2007, pp. 857-866.
2. A. Girgensohn, J. Boreczky, and L. Wilcos, "Key-frame-Based User Interfaces for Digital Video," *Computer*, 2001, vol. 34, no. 9, pp. 61-67.
3. H. Lee, A.F. Smeaton, and J. Furner, "User Interface Issues for Browsing Digital Video," *Proc. 21st Ann. Colloquium on IR Research (ISRG)*, British Computer Soc., 1999.
4. Y. Li, T. Zhang, and D. Tretter. *An Overview of Video Abstraction Techniques*, tech. report HPL-2001-191, HP Laboratories Palo Alto, 2001.
5. Y. Van Houten, M. Van Stetten, and E. Oltmans. *Video Browsing and Summarization: User Interaction Techniques for Video Browsing and Summarization*, pub. no. TI/RS/2000/163, Telematica Instituut, 2000; <http://www.telin.nl/index.cfm?type=doc&handle=12409&language=en>.
6. Y. Van Houten, *A Framework for Video Content Browsing*, pub. no. TI/RS/2001/068, Telematica Instituut, 2001; <http://www.telin.nl/index.cfm?type=doc&handle=19342&language=en>.
7. W. Hürst, G. Götz, and M. Welte, "Interactive Video Browsing on Mobile Devices," *Proc. 15th Ann. ACM Int'l Conf. Multimedia*, ACM Press, 2007, pp. 247-256.
8. W. Hürst, "Interactive Audio-Visual Video Browsing," *Proc. 14th Ann. ACM Int'l Conf. Multimedia*, ACM Press, 2006, pp. 675-678.
9. W. Hürst and G. Götz, "Interface Designs for Pen-Based Mobile Video Browsing," *Proc. 7th Conf. Designing Interactive Systems*, ACM Press, 2008, pp. 395-404.
10. W. Hürst and K. Meier, "Mobile Video Browsing Interfaces," to be published in *Proc. 15th Ann. ACM Int'l Conf. Multimedia*, ACM Press, 2008.
11. T. Masui, K. Kashiwagi, and G.R. Borden IV, "Elastic Graphical Interfaces for Precise Data Manipulation," *Conf. Companion on Human Factors in Computing Systems*, ACM Press, 1995, pp. 143-144.
12. W. Hürst, G. Götz, and T. Lauer, "New Methods for Visual Information Seeking through Video Browsing," *Proc. 8th Int'l Conf. Information Visualisation*, IEEE CS Press, 2004, pp. 450-455.

Contact author Wolfgang Hürst at [huerst@cs.uu.nl](mailto:huerst@cs.uu.nl).

Contact editor Qibin Sun at [qibin.sun@ieee.org](mailto:qibin.sun@ieee.org).