Interactive Manipulation of Replay Speed While Listening to Speech Recordings

Wolfgang Hürst, Tobias Lauer, Georg Götz
Institut für Informatik, Albert-Ludwigs-Universität Freiburg
D-79110 Freiburg, Germany
{huerst,lauer}@informatik.uni-freiburg.de, mail@georg-goetz.de

ABSTRACT
Today's interfaces for time-scaled audio replay have limitations especially regarding highly interactive tasks such as skimming and searching, which require quick temporary speed changes. Motivated by this shortcoming, we introduce a new interaction technique for speech skimming based on the so called rubber-band metaphor. We propose an "elastic" audio slider which is especially useful for temporary manipulation of replay speed and which integrates seamlessly into standard interface designs. The feasibility of this concept is proven by an initial user study.

Categories and Subject Descriptors
H.5.1 [**Information Interfaces and Presentation (e.g. HCI)**]: Multimedia Information Systems – *Audio input/output*

General Terms
Design, Human Factors

Keywords
UI metaphors, multimedia interaction, speech skimming, elastic interfaces, speech interfaces, time-scaled speech replay

1. INTRODUCTION
With the growing ubiquity of speech recordings an ultimate goal related to speech interface research is to make audio files as easy to browse and scan as printed text [2]. One straightforward approach for speech skimming is time-compressed replay, where the speech signal is played at a higher but still comprehensible replay rate. Pitch and timbre preserving techniques, such as the SOLA (*Synchronized Overlap-Add*) algorithm [7], are used to avoid the typical cartoon character voices resulting from simple re-sampling. With such techniques, faster replay becomes a powerful feature for speech skimming. Studies have shown that speech, if modified this way, stays comprehensible even if replayed as fast as 1.6 to 1.8 times normal replay speed [3]. In addition, if the task is not to completely understand the content but just to identify the overall topic, reasonable speed-up factors can be as high as 2.5 to 3 [2]. Users can take advantage of this in order to skim a speech recording, for example, when searching for particular information, skipping parts of minor interest, and so on.

MM'04, October 10–16, 2004, New York, New York, USA.

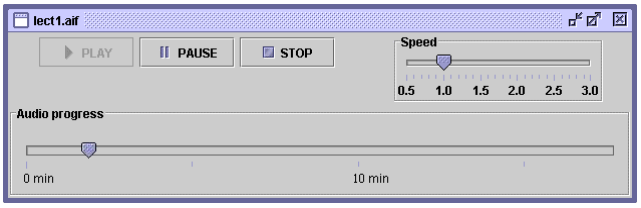

Figure 1. Audio player interface with audio progress bar ("*Audio progress*") and speed control slider ("*Speed*").

Amir et al. [1] describe a scenario where even slower audio replay, i.e., time expansion instead of compression, can make sense. In their study, optimal comprehension was achieved at 0.84 times normal replay rate in a situation where non-native speakers listened to lecture recordings. The benefits of such time-scale modified audio replay have apparently been recognized by the software industry as well, as more and more standard media players offer the option to adjust and modify replay speed [8].

Audio player interfaces typically contain some sort of progress bar (Figure 1, bottom) which represents the current replay position and can be used to randomly access any part of the file. Time-scaling functionality is usually provided through an additional, slider-like controller widget (Figure 1, top right) which allows modification of the replay speed within reasonable ranges. However, such a controller interface has disadvantages especially if frequent changes of replay speed are required. Maybe the most critical issue is that choosing the best speed in a particular situation is not easy. Users often have problems associating the numbers given on the scale with the actual resulting audio feedback. [1] report on initial evidence suggesting that the cognitive perception of time-compressed audio is logarithmic rather than linear in the speed-up factor. In addition, "optimal" replay speed also depends on the document and may even change within a single file. For example, lecturers tend to speak very slowly while writing on a blackboard and speaking at the same time. Hence, temporary speed-up can be useful, even in cases where the file is generally replayed at a slower than normal replay rate, such as described above. Such frequent changes of the replay speed require a user to continuously set (and re-set) the replay rate to different values. The involved interaction is often considered inconvenient and not very intuitive. For example, if the aim is to skip a part of minor interest, it would be much easier if one could just grab the thumb of the audio progress bar and drag it along the scale till the content becomes more important again. However, while the user is moving the thumb, audio feedback is usually paused because providing meaningful audio output in such a situation is critical and sometimes impossible as we will discuss in the next section. In the remainder of this paper we introduce a new interaction concept which enables time-scaled audio

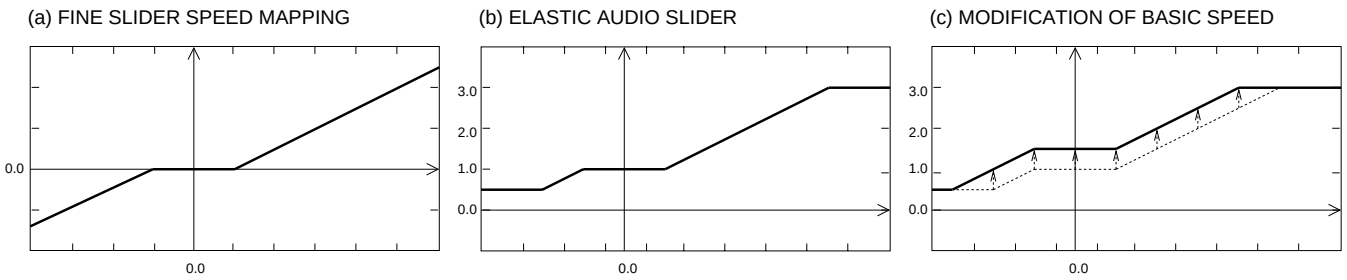

Figure 3. Distance-speed-mappings for (a) the FineSlider (cf. Section 2.1), (b) the elastic audio skimming approach (cf. Section 2.2), and (c) the revised interface design where the mapping is coupled with the speed controller interface (cf. Section 3).

feedback while moving the thumb of a standard progress bar, thus complementing the common speech controller interface design and offering an additional, comfortable way for speech skimming.

2. ELASTIC SPEECH SKIMMING

If a media player supports real-time random access, individual video frames can be immediately displayed while the user drags the slider thumb of the player's progress bar. This kind of navigation, sometimes called *visible scrolling*, is a very intuitive and convenient way to visually browse a document. Replaying individual speech samples in the same way while the user is dragging the slider thumb would result in unintelligible sounds although both digital audio and video consist of individual samples or frames, which are normally played at a fixed rate. The reason for this is one fundamental difference between video and audio: while an individual video frame still carries a meaning, speech samples make sense only when played continuously together with at least some of their neighboring samples. Hence, comprehensible audio feedback while moving the slider thumb of a progress bar is only possible by playing short speech snippets or by using some sort of time-scaling. However, in the first case, synchronization between thumb movements and audio feedback is lost, and in the second case, the required speed-up (or slow-down) factor depends on the continuously changing movements of the slider thumb which are effected by the user and thus cannot be controlled or projected by the sound player. In addition, those movements are sometimes rather jerky and can be too fast or too slow to provide reasonable audio feedback. Fortunately, there are interaction techniques for visual data browsing which enable us to specify restrictions on the thumb movements in order to realize audio feedback in such situations, as we describe in the following.

2.1 Elastic Interfaces for Visible Scrolling

Visible scrolling, i.e., moving the slider thumb while the frame corresponding to the current position is displayed immediately, is very useful and intuitive for visual data browsing [4]. However, it has one significant problem because sliders are limited by window size and screen resolution and therefore do not scale well to large document lengths. Hence, if a long document is mapped to a small slider scale, significant portions of the content might be dropped when moving the slider thumb, even if it is just moved by one pixel. This *scaling problem* has been well studied in the context of scrollbars and static, time-independent documents such as text files. One solution proposed to solve it for static data is Masui et al.'s concept of *elastic interfaces* [5]. It was implemented in the so called *FineSlider* whose basic idea is not to move or drag the slider thumb directly but instead to pull it along a straight line which connects the mouse pointer or cursor with the thumb. The speed at which the thumb follows the cursor movements depends

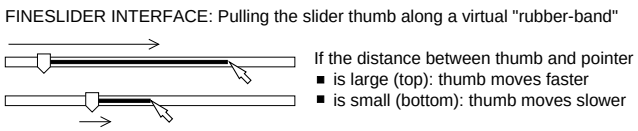

Figure 2. The FineSlider Interface where scrolling speed depends on the distance *d* between thumb and cursor.

on the distance *d* between the cursor and the thumb on the slider bar: if *d* is large, the thumb moves faster, if it gets smaller, the thumb slows down (see Figure 2). This behavior can be illustrated with the *rubber-band metaphor*, where the direct connection between the thumb and the cursor is interpreted as an elastic band: if the cursor is moved away from the thumb, the tension on the rubber-band gets stronger, thus pulling the thumb faster towards the cursor position. If thumb and cursor get closer to each other, the rubber-band loosens, thus reducing the force pulling the thumb, so its movement becomes slower. Generally, the distance *d* between mouse pointer and thumb along a regular slider bar is mapped to movements (i.e., speed and direction) of the slider thumb using a linear mapping function $s(d)$ (cf. Figure 3a).

The main advantage of the FineSlider is that it allows users to access any position within a document independent of the actual length of the file or the size of the corresponding slider scale. By defining the distance-to-speed function *s* appropriately, small distances between the slider thumb and the mouse pointer can be mapped to scrolling speeds which would otherwise only be possible with sub-pixel movements of the thumb, thus supporting access to any random position of the file. While the concept of elastic interfaces was originally introduced for static, time-independent data, we showed in [4] that it can be successfully applied to continuous, time-dependent visual data streams, such as video, allowing the use of a slider to access any individual frame of the video directly. In the following, we describe how elastic skimming can be transferred to the domain of sound data if certain restrictions are considered, despite the fundamental differences between these two media types.

2.2 Elastic Audio Slider for Speech Skimming

As discussed in the introduction of this section, there are several constraints which prevent us from enabling audio feedback while moving the thumb of an audio progress bar: single samples must not be skipped but time-scaled replay should be used. In addition, there must be upper and lower bounds for the replay speed. All these requirements can be met with the elastic interface approach discussed in the previous subsection on visual data browsing. One important side effect of the FineSlider discussed above is that it smoothes the (otherwise jerky) movement of the slider thumb and thus the progress of the document. No matter how fast or slow the

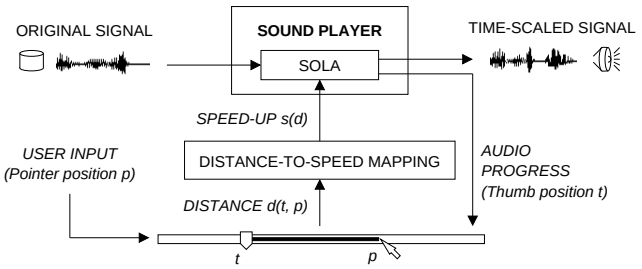

Figure 4. Communication between the elastic slider interface and the sound player.

user is pulling the slider thumb, there are no abrupt jumps in its movements. Changes of its position always happen continuously, following the speed resulting from the distance-to-speed mapping illustrated in Figure 3a. This is because not the *position* but only the *scrolling speed* is directly manipulated. By restricting the distance-to-speed mapping to a lower and upper value once a particular replay speed has been reached, we are able to keep audio feedback within borders that are empirically established to provide intelligible speech feedback. The most critical point when transferring the concept of an elastic slider from visual to acoustic data is that in an audio stream there is no static state which is comparable to a still picture, as described above. However, since the main idea of an elastic speech slider is to temporarily speed up or slow down replay, normal replay can be seen as a "basic state" in case of audio. Hence, moving the mouse pointer to the right of the current thumb position will increase replay speed. Moving it to the left will slow down replay rather than play backwards, which makes sense for visual data streams but would result in meaningless, incomprehensible sound if applied to audio. All these restrictions result in a redefined distance-to-speed mapping *s*, which is illustrated in Figure 3b. It should be noted that in spite of this modification of the function *s*, the approach still conforms to the rubber-band metaphor: the difference is that the force of the band now pulls at the slider thumb moving at regular speed (instead of a static thumb). When dragged to the right, the rubber-band accelerates the thumb, depending on how far it is stretched. If it is pulled to the left of the thumb, it works as a brake, slowing down the thumb. Again, the force affecting the thumb movement increases with the distance, i.e., the tension on the rubber-band.

The implementation of this functionality is illustrated in Figure 4. Whenever the elastic slider function is activated, the horizontal distance *d* between the mouse pointer and the slider thumb is measured continuously and mapped to the corresponding speed-up factor *s*(*d*), which is fed to the sound player containing the audio time-scaling algorithm. The algorithm adjusts audio replay to the given speed-up factor *s*(*d*). One technical prerequisite is that the time-scaling algorithm must run in real time, so any speed changes become effective immediately. Following He and Gupta [3], who point out that the better audio quality of sophisticated algorithms hardly outweighs their computational complexity, we have integrated a real-time variant of the SOLA algorithm [7], which provides good results while still being computationally simple. The time-scaled sound data are sent to the audio output. In addition, the sound player always feeds the current audio progress back to the slider bar, which in turn updates the position of the slider thumb accordingly. As soon as the elastic function is deactivated, replay reverts to normal speed. Figure 5 shows the visualization of the actual implementation. When the mouse pointer moves over the slider scale, the areas for faster, slower,

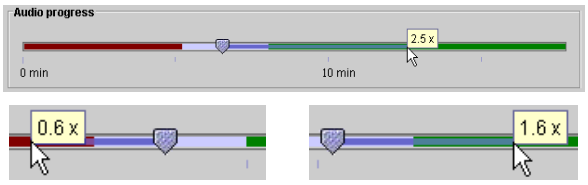

Figure 5. Visualization of the elastic audio slider.

and normal replay are indicated through different colors in the slider bar. When pulling the thumb towards the pointers direction, the virtual rubber-band connecting thumb and cursor is indicated as well as a tool-tip which shows the actual replay speed.

For the integration of the skimming functionality into the existing interface design, we experimented with two different approaches. In the first version, the elastic functionality is activated by holding down the mouse button directly on the slider thumb and dragging it to the left or right (see Figure 6a). In addition, users are still able to access any random position directly by clicking at the corresponding position in the slider bar ("*snap-to-click*"). The second approach (Figure 6b) reverses those functions. Grabbing and dragging the slider thumb directly works like it does in traditional interfaces: audio is paused until the mouse button is released at the target position. Pressing the mouse button anywhere on the slider bar (except for the thumb) activates elastic skimming. The speed is set according to the distance, as described above, and dragging from there increases or decreases replay speed depending on the direction of movement. In the following we describe a qualitative usability study where these two options are compared and the overall usability of the proposed approach is verified with a heuristic evaluation.

3. USABILITY AND REVISED DESIGN

Heuristic evaluation as proposed by Nielsen [6] is a well established HCI method for qualitative usability evaluation especially in early design stages. It requires as few as five users in order to verify the feasibility of an interaction concept and to identify problems and flaws in the implemented design. Hence, we presented our implementation to five test users (who had not seen or used the interface before) in order to evaluate it based on the set of usability related heuristics proposed in [6]. While overall feedback was very positive, some important observations were made and a few problems in the design were identified.

The first integration scheme (see Figure 6a) did not appear to be useful because it requires users to grab the thumb while it is moving during normal replay. The need to hit such a moving target puts a high cognitive load on the users and makes this approach not useful in practical applications. While the second version (Figure 6b) worked quite well, some users complained that the elastic skimming functionality replaced the possibility to jump directly to a specific position on the progress bar. This functionality can be very useful and important, for example, if one wants to go back a few seconds in order to re-listen to the last sentence. As a consequence we changed the integration in a way as illustrated in Figure 6c: a click on the scale (with an immediate release of the mouse button) makes the thumb jump to this position. Elastic skimming is only evoked when the user keeps holding the mouse button and moves the pointer to the left or right. Grabbing the thumb directly results in the same behavior as in Figure 6b. With this approach, all possibilities usually known from audio progress bars are still provided and elastic audio

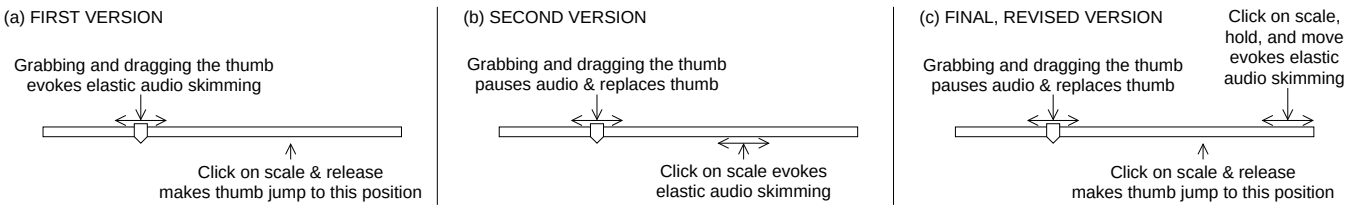

Figure 6. Different approaches for the integration of the elastic audio slider functionality into the progress bar.

skimming integrates seamlessly into the traditional user interface design. A minor flaw discovered by the test users was that elastic skimming could not be applied once the thumb was near the end of the slider scale. Hence, we adapted the implementation in a way such that mouse events are still perceived and processed by the system when the pointer is dragged out of the slider window.

Beside these two problems, all users agreed that the proposed approach for elastic speech skimming is feasible, useful, easy to handle, and improves the overall browsing experience significantly. They found it particular useful for short, temporary speed-ups (or slow-downs), while they agreed that for continuous speed changes, i.e., when the aim is to listen to a file at a different but fixed replay rate for a longer amount of time, the speed controller interface is the better choice. This is because the thumb is continuously moving and therefore influencing the replay speed which makes it hard to keep replay at a fixed value with the elastic slider. However, some users said that the possibility to speed-up replay temporarily is so important and can be done with the elastic slider so easily that they would also like to have this opportunity when continuously listening to a file with a replay rate other than normal. To fulfill this need, we coupled the functionality offered by the speed controller interface with the possibility for temporary speed-up using the elastic slider as follows: modifying the replay speed with the controller interface adapts the distance-to-speed mapping as illustrated in Figure 3c, i.e., the speed selected with the speed controller (1.5 in the example in Figure 3c) serves as a new basic speed for the elastic slider and the behavior to the left and right of the neutral area around the slider thumb is adapted accordingly. The test users particularly liked this feature when we re-presented the revised interface design to them. In addition, such a behavior fits well to observations made in a study by [1], where different *natural speeds* were identified for varying speech documents. The natural speed of a file is defined as the rate that listeners, on average, agree to be the best for comprehension for this particular file. By coupling the elastic slider with the standard speed controller, we enable users to select their preferred replay rate, which is then used as the new basic speed. Once a part of minor (or particular) interest is identified, the elastic slider can be used for a temporary speed-up (or slow-down, respectively) until the mouse button is released and replay switches back immediately to the previously selected normal speed.

Based on the qualitative user study, we can conclude that the elastic audio slider approach proposed in this paper proved to be feasible, easy to handle and useful for interactive manipulation of replay speed and speech skimming tasks. It fits well into existing and established user interface designs for common audio players and complements their functionality in particular for situations where temporary modifications of replay speed are required. Our current work includes a quantitative evaluation in order to back up the initial observations gained with the qualitative usability study.

In addition, we aim at integrating additional time-compression techniques, such as pause reduction, into the system. Maybe the most challenging and exciting opportunity for future research will be the combination of the elastic audio skimming functionality with the opportunity to skim visual data streams at the same time, thus providing real multimodal data browsing. While we have already shown the usefulness of elastic browsing for video data [4], combined audio-visual browsing raises a whole range of new questions regarding issues such as how to handle the different upper and lower speed-up bounds for audio and video, whether (and how) to provide audio feedback during visual skimming in reverse direction, or how to maintain synchronized replay if pause reduction is used.

4. ACKNOWLEDGMENTS

This work was supported by the German Research Foundation (DFG) as part of its research initiatives “Distributed processing and exchange of digital documents (V3D2)” and “Net-based knowledge communication in groups”.

5. REFERENCES

[1] Amir, A., Ponceleon, D., Blanchard, B., Petkovic, D., Srinivasan, S. and Cohen, G. Using Audio Time Scale Modification for Video Browsing. In *Proceedings of the 33rd Hawaii International Conference on System Sciences (HICCS 2000)*, Maui, Hawaii, USA, January 2000.

[2] Arons, B. SpeechSkimmer: A System for Interactively Skimming Recorded Speech. In *ACM Transactions on Computer Human Interaction*, 4(1), 3-38, March 1997.

[3] He, L. and Gupta, A. Exploring Benefits of Non-Linear Time Compression”. In *Proceedings of the 9th ACM International Conference on Multimedia*, Ottawa, Canada, Sept. 2001.

[4] Hürst, W., Götz, G. and Lauer, T. New methods for visual information seeking through video browsing. In *Proceedings of the 8th International Conference on Information Visualisation (IV 2004)*, London, UK, July 2004.

[5] Masui, T., Kashiwagi, K., and Borden IV, G.R. Elastic graphical interfaces for precise data manipulation. In *Conference companion on Human factors in computing systems*, ACM Press, pp. 143-144, 1995.

[6] Nielsen, J. and Mack, R. L. (eds.) *Usability Inspection Methods*, J. Wiley & Sons, New York, NY, USA, 1994.

[7] Roucus, S. and Wilgus, A. High quality time-scale modification for speech. In *IEEE International Conference on Acoustics, Speech, and Signal Processing, Vol. 2*, 1985.

[8] Windows Media 9 Series, Fast and Flexible Playback: http://www.microsoft.com/windows/windowsmedia/9series/player/fast.aspx