Audio Engineering Society

Convention Paper

Presented at the 131st Convention
2011 October 20–23 New York, USA

A Nimble Video Editor that Puts Audio First

Jörn Loviscach[1]

[1] *Fachhochschule Bielefeld (University of Applied Sciences), 33602 Bielefeld, Germany*

Correspondence should be addressed to Jörn Loviscach (joern.loviscach@fh-bielefeld.de)

ABSTRACT

Video editing software tends to be feature-laden, to respond sluggishly to user input—and to be focused on the visuals rather than on auditory aspects. All of this is a burden when the task is to edit material in which audio plays the leading role, such as a talk show, a video podcast, or a lecture recording. This work presents a highly visual, no-frills video editor that is tailored to these tasks. It combines a range of techniques that speed up the process of searching and reviewing. They range from an overview-and-detail display to speech recognition to constant-pitch variable-speed playback . The implementation is heavily multithreaded and fully leverages the computer's main memory to ensure a highly fluid interaction.

1. INTRODUCTION

Video footage that merely accompanies an audio track poses a major headache in editing. Standard video editing software is focused on the visuals. These, however, are meager when it comes to talking heads as seen in talk shows and recorded lectures. For such material, the usual filmstrip-plus-waveform display conveys little information about what happens when. Reviewing the material and finding the right spots for editing become time-consuming tasks.

This work improves on this by leveraging audio processing, data visualization, multi-core computing and by streamlining the user's interaction with the software. Aimed at cleaning up video podcasts and lecture videos, the software focuses on editing through deletion of parts. This is specifically aimed at facilitating the production of video blogs and lecture videos.

Every function is available at every time, even during playback. The user can already edit the video while three types of data are being extracted from the video file for visualization: single frames for the overview of the images contained in the video; the audio envelope and the zero-crossing rate for the visualization of the audio track; and text with timing information for a transcript.

In standard video editing software, easily half of the screen can be filled with controls or can even be

empty, such as showing blank tracks. In contrast to that, in the video editor demonstrated here almost the entire screen is filled with an overview-and-detail display similar to that of Casares et al. [3]: One "lane" shows the complete length of the edited video, another "lane" show the zoomed-in view of a selected part. Improving upon the work of Casares et al., the text transcript is placed between the overview track and the detail track to save space; furthermore, the words are tilted if needed to allow for more words to be displayed. If there still are too many words, only a fraction of them is shown. The speech recognizer's confidence value is displayed through the shade of the lettering.

Through clicking and dragging, the user determines the position of the playhead, the range to be shown in the detail lane, and the ranges to be deleted, see Fig. 1. There is no menu and there are no tools to be picked. A loop function enables incremental adjustments while listening to the result. For a quick review or for a detailed analysis, the playback speed can be adjusted while maintaining constant pitch.

This paper is structured as follows: Section 2 outlines related work. The visualization of the visual content of the video is covered by Section 3, the visualization of the audio signal by Section 4. Section 5 describes the speech recognition and its graphical representation. The implementation of the prototype is detailed in Section 6. Section 7 concludes the paper and provides an outlook.

2. RELATED WORK

Reviewing video for searching and/or editing is a time-consuming process. Doing this by watching the video at its original speed often is prohibitively slow. This problem has spurred a range of research. One prominent proposal is the SILVER system by Casares [3] et al., which—as one of its many features—presents increasingly finer portions of the timeline in parallel. This can be viewed as focus and context in the spirit of general zooming user interfaces [4].

Displaying the video as a filmstrip has become the standard choice in editing software, even though a plethora of methods has been introduced to summarize (dynamic) video through (static) images or sequences of images. For instance, one can extract keyframes [21], arrange them in 2D [22] or form seamless and zoomable collages along the timeline [2, 8]. Tang et al. [11] employ the artistic technique [9] of slit scanning to show regions and patterns in weeks of video. They, too, apply a zoom hierarchy similar to [3]. In subsequent work, Tang et al. [17] demonstrate a generalized version in which the "slit" no longer is a vertical line.

A straightforward way to speed up video browsing and editing is to accelerate the playback. To keep speech intelligible, the inevitable raise in pitch needs to be corrected. Then, novice users can tolerate a speedup factor of 1.7 without detrimental effects to comprehension, as Amir et al. [5] report.

Audio editing alone, too, is ridden with the problem of time-consuming searching and reviewing. Already 30 years ago, Schmandt [16] introduced a "digital dictaphone" that employed visualization and speech recognition—to the degree possible at the time.

Many visual solutions are known that are more expressive than the common waveform and spectrogram displays. Audio signals may for instance be visualized through bands of colored stripes generated from psychoacoustic parameters [7, 14] or through synthetic low-frequency waveforms [13]. The Comparisonics method colors waveforms according to the currently dominant frequency and amplitude variance [6]. For further methods see [10].

3. VIDEO DISPLAY

To quickly generate an increasingly precise overview of the image content, frames of the video are extracted at temporal positions gained through progressive bisection. Each of these temporal positions is rounded to that of the nearest previous frame that is stored as keyframe in the video file, as long as that keyframe has not already been extracted. This way of preferring keyframes speeds up the decompression approximately by one quarter, which allows showing more details earlier. In addition, important changes tend to be accompanied by keyframes and hence show up early in the process. On the screen, the display of the frames is updated regularly to reflect the ongoing extraction.

The extracted images are stored in RAM at reduced resolution (e. g. 400×225 pixels). Furthermore, a compression tailored to this application is employed. Given a certain column of pixels of some freshly extracted frame, the system only rarely stores these

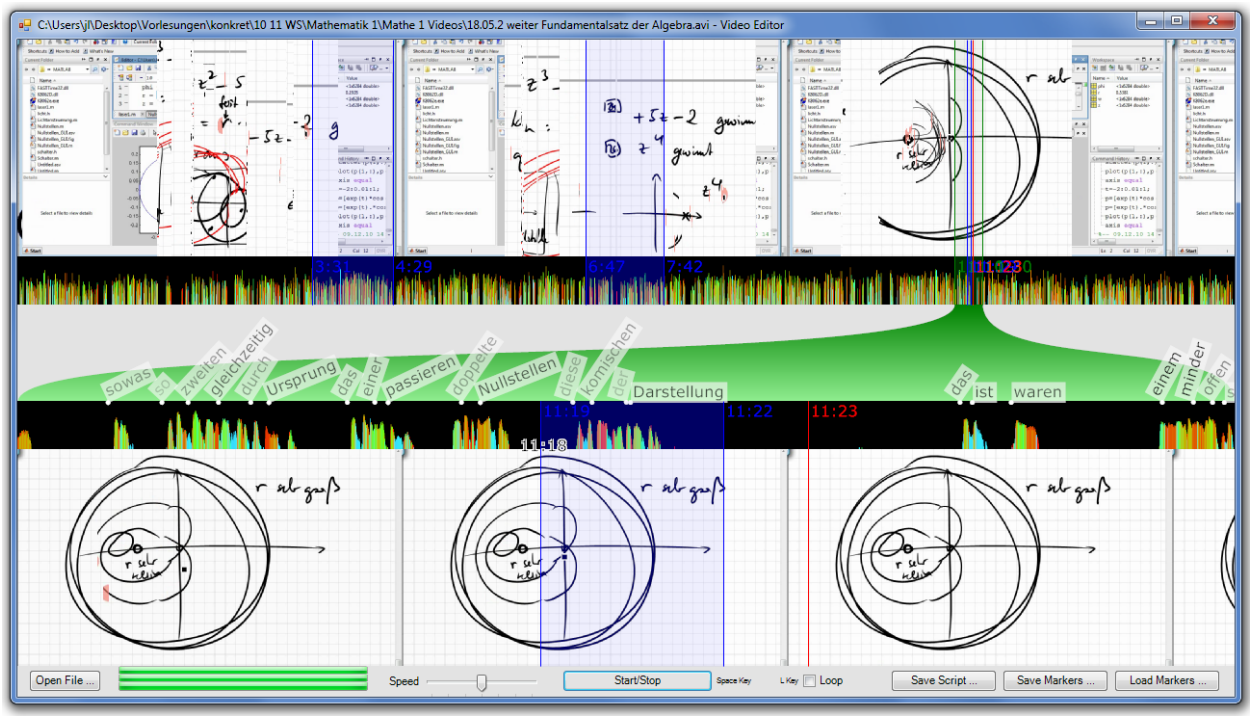

Fig. 1: A wealth of visualization techniques ranging from colored waveforms to a synchronized transcript speeds up the operation. The parts of the video to be deleted are marked with a blue tint.

RGB data. If it rather finds that the RGB data it stored for the same column before or after that frame are highly similar, it only stores a pointer to these. This is particularly efficient for video captured from a computer's screen, where 10,000 frames—that is, 2.7 GByte of raw RGB at the reduced resolution—fit into 100 to 300 MBytes of RAM. The amount of 1 GByte of RAM is set as an upper limit. For longer videos, an increasing number of frames will be left out.

The video is not represented as the usual film strip. Instead, the visualization combines the film strip with a slit-scan effect like Tang et al. [11]. Each column of pixels on the screen is fetched as one column of pixels from one of the extracted video frames, which leaves open two questions: Which frame? And which column?

The frame from which the column is taken is the frame that belongs to that precise position of the timeline (horizontal position of the column measured in pixels times the total number of frames divided by the width of the filmstrip). Hence, global changes are visible with extreme temporal resolution, see Fig. 2.

The pixel column that is taken from this frame is the column that would be used for a standard film strip display. Hence, for static video the standard look of a film strip is retained. This makes it easier to understand the content, in particular when zooming in.

4. AUDIO SIGNAL DISPLAY

The audio waveform is analyzed to find a smoothed power envelope and a smoothed zero-crossing rate. The zero crossing rate represents an estimate of twice the most prominent frequency in a signal, in particular for a monophonic source such as a recorded voice. Both the power envelope and the zero crossing rate are stored in RAM at a resolution of approximately 5 ms.

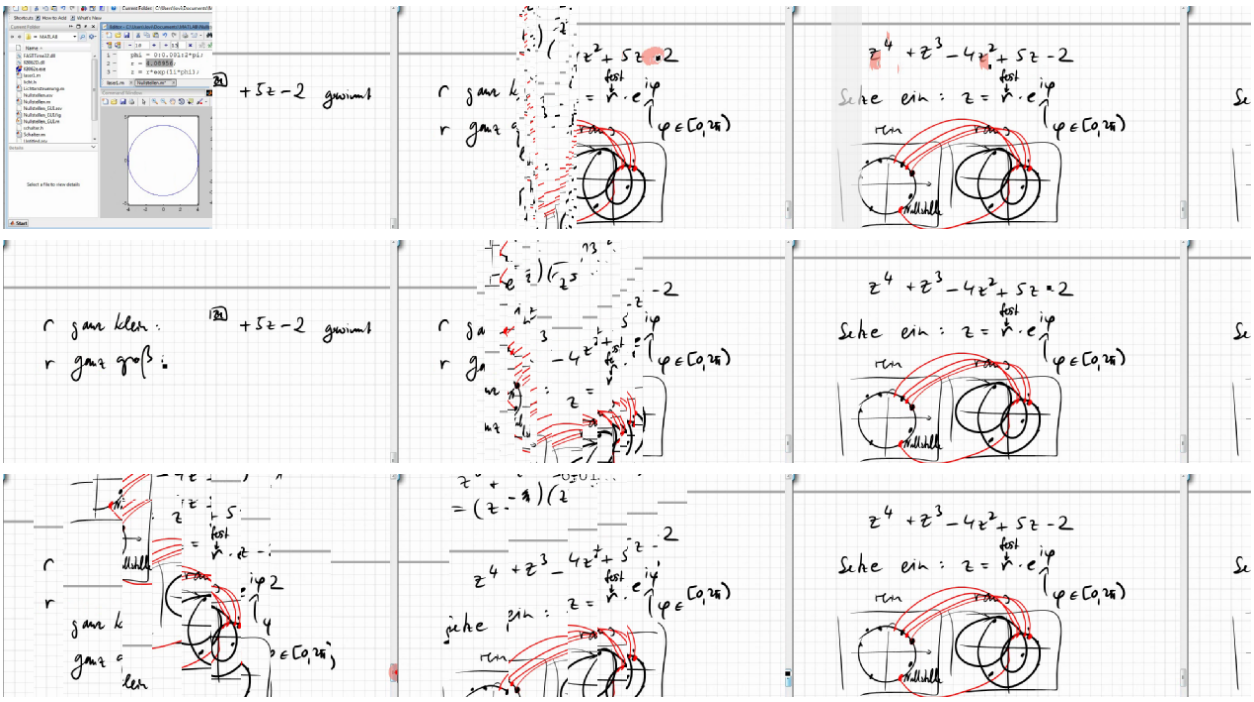

Fig. 2: At a coarse zoom level (top), the overall structure of the video becomes apparent with high temporal resolution. As the zoom level gets finer (middle) and finer (bottom), the display increasingly resembles the classic filmstrip visualization. Note how the V-shaped display of scrolling down and up in the recorded video unfolds in different zoom levels.

To stay close to human perception, the standard waveform extending up and down from a zero line is replaced by a one-sided diagram showing the logarithm of the power, see Fig. 3.

Using the zero-crossing rate $zcr(t)$, the display is colored to enable quickly spotting regions of different frequency content, such as telling "s" from "oo". In contrast to Comparisonics waveform coloring [15], the mapping to colors is adjusted to best capture the actual data range of the audio track at hand. To this end, the mean m and the standard deviation s of the zero-crossing rate are determined. The visualization is fed with data normalized accordingly, namely the z-score $\frac{zcr(t)-m}{s}$.

For a better match with the soft edges found in today's graphical user interfaces, the resulting visualization of the audio signal is antialiased by eightfold horizontal supersampling.

5. SPEECH RECOGNITION

The audio track is also sent to the built-in speech recognizer of Microsoft Windows, which outputs text, confidence, and timing. As with all dictation systems, the reliability of this recognizer is limited, even when trained to the presenter's voice. Nonetheless, one can quickly guess which words were actually meant; in addition, conspicuous technical terms tend to be recognized accurately.

Current video editing software such as Adobe Premiere, Apple Final Cut, and Avid Media Composer can already employ speech recognition or phoneme-to-script alignment, at least through add-ons. In most cases, however, this is intended for searching. If a transcript is generated at all, such as in Adobe Premiere, it is shown in a separate window, not in conjunction with the filmstrip—unlike the solution of Casares et al. [3]. A transcript in a separate win-

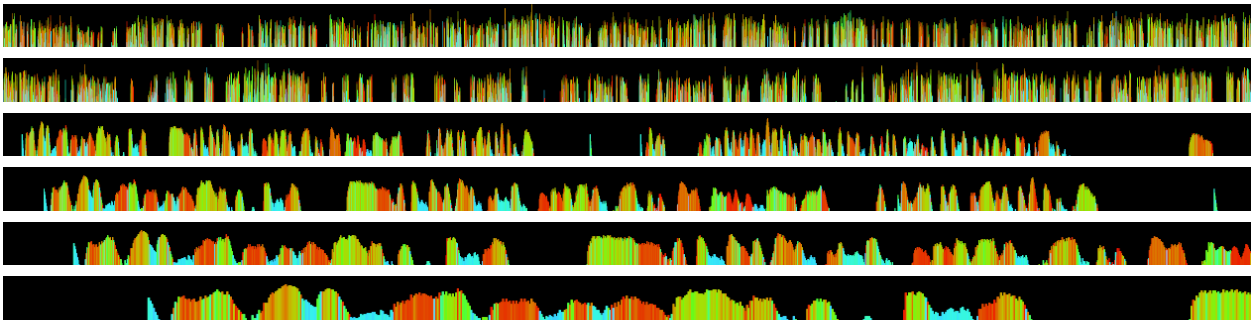

Fig. 3: The audio signal—six different zoom levels shown—is displayed through its power envelope, colored according to its zero crossing rate (low/mid/high frequencies: red/green/blue). To fully exhaust the range of color, the actual content of the current track is used for normalization.

dow does not allow setting the timing of edit points precisely.

The video editor presented here keeps the transcript adjacent to the visual content, see Fig. 1. The start time of each word is aligned with the timeline, as marked through a white dot. For coarse zoom levels, the words are tilted as much as needed, see Fig. 4. This is intended to retain as much legibility as possible. In an earlier version of the software, the font size was reduced if needed, which quickly led to overly small type. If the text gets more crowded than can be catered for by the tilting, the system starts to drop words.

To indicate the reliability estimate given by the speech recognizer, the words are rendered in a stronger or a lighter tone. This is one of the many places in which this user interface makes use of just noticeable differences to preserve a hierarchy of information [18].

6. IMPLEMENTATION

The prototype has been developed exclusively in the language C#, utilizing Microsoft .NET. The software is heavily multithreaded, delegating audio and video extraction and visualization as well as the rendering to the screen after updates or zoom events to separate threads, see Fig. 5. This ensures responsive interaction at all times.

Existing software is reused to a large degree: First, the speech recognition employs Microsoft Windows' built-in recognizer. Currently, Windows 7 in its Professional variant does not, however, allow extending its speech recognizer to other languages. One has to stick to the language it has been delivered in. This is the reason why the screenshots in this paper show German texts. Another issue with the speech recognizer turned out to be that it does not report the word timing correctly for all sampling rates. In our experiments, a sampling of 22.05 kHz did work, one of 44.1 kHz didn't.

Second, the audio playback is handled through Windows Media Player, used as an invisible ActiveX component of the type `AxWMPLib.AxWindowsMediaPlayer`. During playback, the temporal position of Windows Media Player is read continuously to position the line showing the "playback head". To jump over regions marked as to be deleted and to create a loop, this position is set from the main program. Windows Media Player has a rather hidden setting to change the playback speed while keeping the original pitch. The quality of this effect is acceptable for speech. The speed slider on the video editor's graphical user interface acts as a remote control for this setting.

Third, the actual production of an edited video file is delegated to the open-source program VirtualDub. The edits created by the user are written to a script file that can be read and executed by VirtualDub. This software proved unbeatably efficient for this task, in particular as it can keep the number of video frames that are decompressed and recompressed at a minimum.

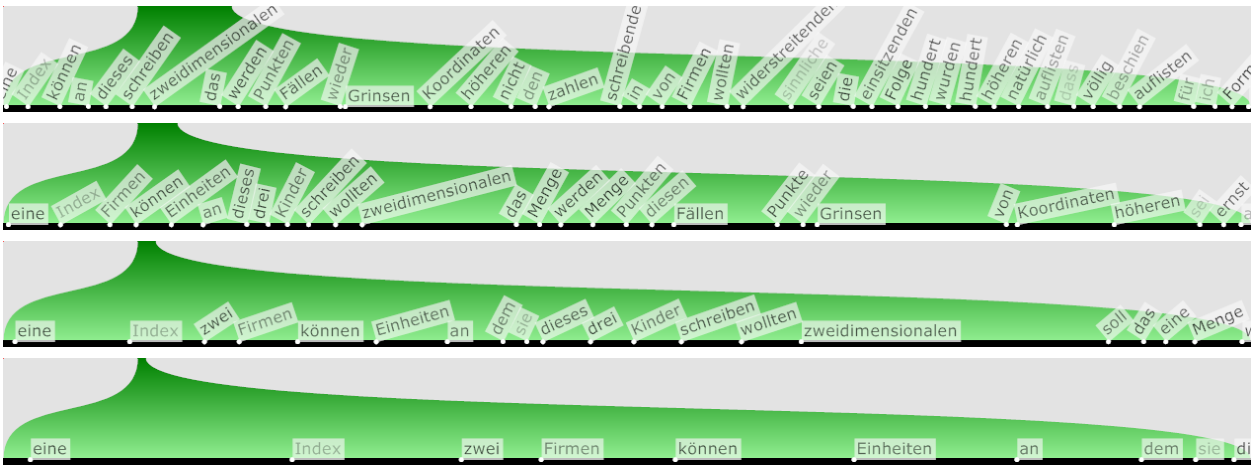

Fig. 4: In fine zoom levels (bottom), the recognized text can be presented horizontally and thus easy to read. The coarser the zoom level gets (middle), the more the words are tilted. Eventually (top) words are left out to not overcrowd the display.

7. CONCLUSION AND OUTLOOK

This work has presented how to combine known and novel methods of signal analysis and data visualization to create a video editor targeted at material in which audio plays a prominent role.

Future work may improve the audio visualization through additional parameters such as mel-frequency cepstral components extracted from the audio track. Important words could automatically be found in the transcript and emphasized graphically [19]. The user could correct errors the automatic speech recognizer has made in the transcript [20]. For "pencast"-style videos produced on a Tablet-PC-type computer, the results of the built-in handwriting recognizer may be used to verify the results of the speech recognizer or may be displayed in addition to the audio transcript. To further speed up searching and reviewing, audio playback could be accelerated non-linearly [1, 12], for instance by jumping over silent passages.

8. REFERENCES

[1] B. Arons. SpeechSkimmer: a system for interactively skimming recorded speech. *ACM Trans. Comput.-Hum. Interact.*, 4:3–38, March 1997.

[2] C. Barnes, D. B. Goldman, E. Shechtman, and A. Finkelstein. Video tapestries with continuous temporal zoom. In *ACM SIGGRAPH 2010 papers*, SIGGRAPH '10, pages 89:1–89:9, New York, NY, USA, 2010. ACM.

[3] J. Casares, A. C. Long, B. Myers, S. Stevens, and A. Corbett. Simplifying video editing with SILVER. In *CHI '02 extended abstracts on human factors in computing systems*, CHI EA '02, pages 672–673, New York, NY, USA, 2002. ACM.

[4] A. Cockburn, A. Karlson, and B. B. Bederson. A review of overview+detail, zooming, and focus+context interfaces. *ACM Comput. Surv.*, 41(1):1–31, 2008.

[5] G. Cohen, A. Amir, D. Ponceleon, B. Blanchard, D. Petkovic, and S. Srinivasan. Using audio time scale modification for video browsing. In *Proceedings of the 33rd Hawaii International Conference on System Sciences – Volume 3*, pages 3046–3055, Washington, DC, USA, 2000. IEEE Computer Society.

[6] Comparisonics Corp. Comparisonics waveform display. `http://www.comparisonics.com/`, last accessed 2011-07-17.

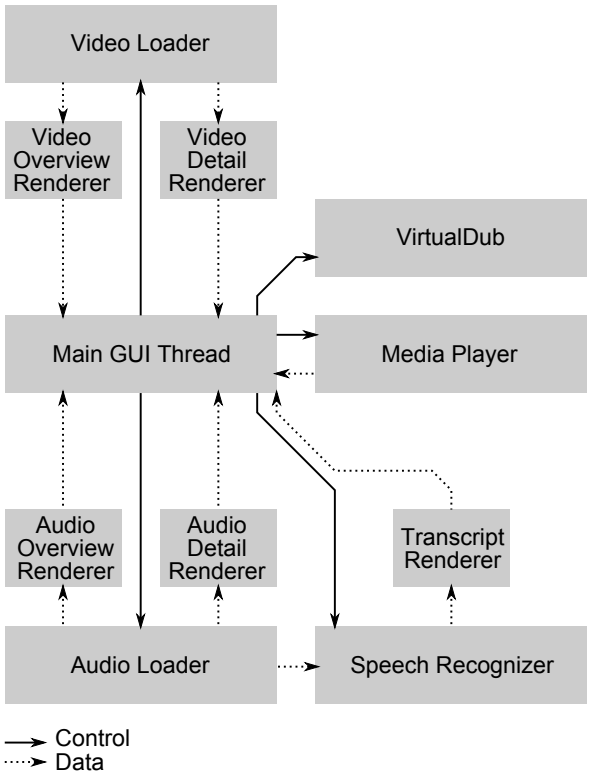

Fig. 5: To enable a fluid interaction, loading (including analyzing) and graphical rendering of the extracted data are delegated to separate threads.

[7] P. R. Cook and G. Tzanetakis. Audio information retrieval (AIR) tools. In *Proceedings of ISMIR 2000*, 2000.

[8] C. D. Correa and K.-L. Ma. Dynamic video narratives. *ACM Trans. Graph.*, 29:88:1–88:9, July 2010.

[9] A. Davidhazy. Slit scan and strip photography overview. `http://people.rit.edu/andpph/text-streak-strip-scanning-imaging-overview.html`, last accessed 2011-07-17, 2010.

[10] K. Gohlke, M. Hlatky, D. Black, S. Heise, and J. Loviscach. Track displays in DAW software: Beyond waveform views. In *Proceedings of the 128th Convention of the AES*, 2010. Paper No. 8145.

[11] S. Greenberg, M. Nunes, S. Carpendale, and C. Gutwin. What did I miss? Visualizing the past through video traces. In *Proceedings of the European Conference on Computer-Supported Cooperative Work*, ECSCW '07, pages 1–20, 2007.

[12] L. He and A. Gupta. Exploring benefits of non-linear time compression. In *Proceedings of the ninth ACM international conference on Multimedia*, MULTIMEDIA '01, pages 382–391, New York, NY, USA, 2001. ACM.

[13] J. Loviscach. The quintessence of a waveform: Focus and context for audio track displays. In *Proceedings of the 130th Convention of the AES*, 2011. Paper No. 8397.

[14] A. Mason, M. Evans, and A. Sheikh. Music information retrieval in broadcasting: Some visual applications. In *Proceedings of the 123rd Convention of the AES*, 2007. Paper No. 7238.

[15] S. V. Rice. Frequency-based coloring of the waveform display to facilitate audio editing and retrieval. In *Proceedings of the 119th Convention of the AES*, 2005. Paper No. 6530.

[16] C. Schmandt. The intelligent ear: A graphical interface to digital audio. In *Proceedings of the IEEE international conference on cybernetics and society*, pages 393–397. IEEE, 1981.

[17] A. Tang, S. Greenberg, and S. Fels. Exploring video streams using slit-tear visualizations. In *Proceedings of the working conference on advanced visual interfaces*, AVI '08, pages 191–198, New York, NY, USA, 2008. ACM.

[18] E. R. Tufte. *The visual display of quantitative information*. Graphics Press, Cheshire, CT, 2nd edition, 2001.

[19] S. Vemuri, P. DeCamp, W. Bender, and C. Schmandt. Improving speech playback using time-compression and speech recognition. In *Proceedings of the SIGCHI conference on human factors in computing systems*, CHI '04, pages 295–302, New York, NY, USA, 2004. ACM.

[20] S. Whittaker and B. Amento. Semantic speech editing. In *Proceedings of the SIGCHI conference on human factors in computing systems*, CHI '04, pages 527–534, New York, NY, USA, 2004. ACM.

[21] C.-I. Wu, C.-M. James Teng, Y.-C. Chen, T.-Y. Lin, H.-H. Chu, and J. Y.-J. Hsu. Point-of-capture archiving and editing of personal experiences from a mobile device. *Personal Ubiquitous Comput.*, 11:235–249, April 2007.

[22] M. M. Yeung and B.-L. Yeo. Video visualization for compact presentation and fast browsing of pictorial content. *IEEE Transactions on Circuits and Systems for Video Technology*, 7(5), 1997.