DiMaß: Audio Scrubbing and Skimming with Continuous, High-Fidelity Feedback

Eric Lee and Jan Borchers
Media Computing Group, RWTH Aachen University, Germany

Abstract

DiMaß is a technique for audio scrubbing and skimming using direct manipulation. Unlike existing techniques, DiMaß is able to render audio with high fidelity and maintain the original audio pitch, but still precisely and smoothly track the user input. DiMaß consists of three parts: motion estimation takes position events from an input device such as a mouse and calculates the desired audio position and velocity; input tracking uses this information to compute an adjusted audio play rate; and time-stretching alters the audio play rate at arbitrary speeds while preserving the original audio pitch with high fidelity. DiMaß is implemented as part of a Cocoa-based audio navigation tool using the following Mac OS X technologies: Quartz 2D for the waveform drawing, the Accelerate framework for high-performance digital signal processing, and Core Audio for audio output. DiMaß enables both content creators and consumers to more easily navigate digital media content.

Introduction

Despite the increasing popularity of digital media content such as downloadable music and podcasts, interfaces for scrubbing and skimming through this content remain quite primitive. When browsing through text, and even movies, one relies on visual "snapshots" of the material to accurately pinpoint the search target. Audio, however, must be interpreted over time to be meaningful, making such snapshots impossible.

DiMaß (**Di**rect **M**anipulation **A**udio **S**crubbing and **S**kimming) is a novel solution for audio scrubbing based on our previous work on a flexible-time media framework for interactive conducting systems [7] and a high-fidelity, pitch-preserving, time-stretching algorithm [4]. Many current audio editing tools, such as Adobe Audition or Apple Logic, already support audio scrubbing; unfortunately, they offer either no audio feedback, low-quality audio feedback, or audio feedback with undesirable pitch-shifting artifacts. DiMaß is the first audio scrubbing technique that renders audio with high fidelity, and maintains the original audio pitch, but still precisely and smoothly tracks the user input.

Objectives

We aim to support audio scrubbing and skimming for the following types of tasks:

- locating a specific point in a music recording or podcast ("what was the model number of that camera reviewed in yesterday's Macworld podcast...?").
- cutting and splicing audio segments to create, for example, a two-minute summary of a thirty-minute interview.

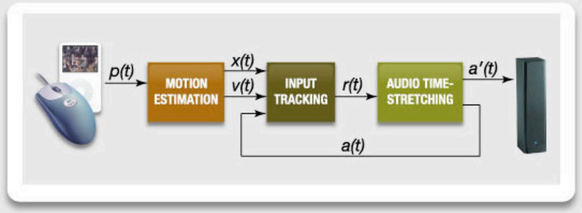
In the design of DiMaß, we adopted the principle of *direct manipulation*, a term used in human-computer interaction to describe the class of interfaces with "visible objects and actions of interest, with rapid, reversible, incremental actions and feedback" [8]. In particular, DiMaß:

- allows users to directly interact with the audio timeline (position); many existing systems for speech skimming, in contrast, utilize rate-based control [1,3].
- responds to user input with low latency, even with low-cost, low-fidelity, positioning devices such as a mouse; while certain devices such as disc jockey (DJ) turntables already provide immediate response, they are also extremely expensive.
- provides continuous audio feedback using a high-fidelity time-stretching algorithm.

Design

DiMaß can be described as users imposing their own sense of time, "user time", onto the audio. We further distinguish between "audio time", the timeline that is embedded in the original audio file, from "real time", as obtained, for example, from a clock. The relationship between audio time and real time dictates how fast the audio is playing at a given moment, for example, and this relationship is often used to study and analyze timing patterns in music [2].

We divide DiMaß into three parts: motion estimation, input tracking, and audio time-stretching.



DiMaß block diagram.

Motion Estimation

The motion estimator receives position events, *p(t)*, and estimates the user's instantaneous position, *x(t)*, and velocity, *v(t)*.

To compensate for the fact that relative positioning devices such as a mouse only report *changes* to position, we use an exponential rise and decay function to estimate the velocity.

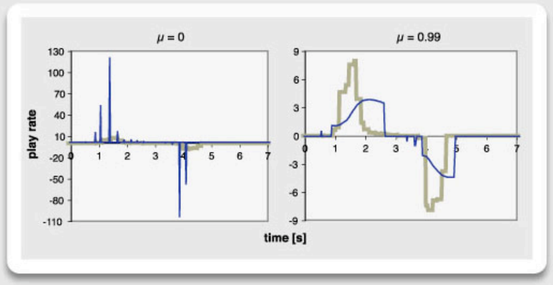
Input Tracking

The input tracker synchronizes the audio play rate, *r(t)*, to *x(t)* and *v(t)*.

Existing synchronization algorithms [5,6,7] are unsuitable for DiMaß, because they assume, for example, that:

- the drift between the independent and the dependent timebases is small.
- there are no sudden changes to the speed and/or position of the user input.
- time is always moving forwards.

Our improved synchronization algorithm combines results from an instantaneous rate adjustment together with a gradual rate adjustment. We also introduce a "viscosity" parameter μ to smoothen the play rate at the expense of a little responsiveness.



Audio Time-Stretching

The last step is to time-stretch the audio using the adjusted play rate to produce *a'(t)*, and return an updated position in the original audio timeline, *a(t)*, back to the input tracker; this feedback is necessary to ensure precise synchronization with the user input [6,7].

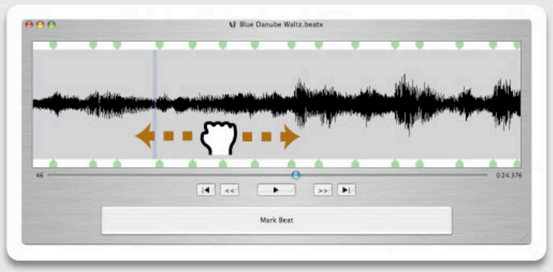
We use a frequency domain algorithm based on the phase vocoder [4] that produces results comparable to the TimePitch audio unit in Core Audio. The algorithm is implemented as part of the Semantic Time Framework, a software library we created for developing multimedia applications with time-based interaction [7]. The time-stretching feature in the Semantic Time Framework is unique in that it supports arbitrary, including backwards, play rates, and it also preserves the original (unstretched) to time-stretched audio time mapping (required by the input tracker for precise synchronization).

Implementation

Beat Tapper

We incorporated DiMaß into *Beat Tapper*, a tool we developed for tagging musical recordings with beat metadata. Using *Beat Tapper*, users mark beats in an audio file by "tapping along" while the audio is playing, and manually fine-aligning them afterwards using a visual representation of the audio waveform. *Beat Tapper* can also play these beats back synchronously with the music as audible "taps".

The audio play rate can be adjusted dynamically, and using DiMaß, users can "grab" on to the audio waveform and shuffle it around using the mouse while receiving continuous audio feedback.



Mac OS X Technologies

We used the following Mac OS X technologies to realize DiMaß/Beat Tapper:

- **QuickTime**: Beat Tapper can import audio from any QuickTime-playable movie using the audio extraction capabilities of QuickTime 7.
- **Accelerate**: The Semantic Time Framework makes extensive use of the Accelerate framework to process audio; the Accelerate framework is also used to pre-compute scaled versions of the audio waveform for display.
- **Quartz 2D**: The waveform is drawn using Quartz 2D to maximize performance.
- **Core Audio**: The output audio streams are played through Core Audio.

Beat Tapper is a document-based Cocoa application that runs natively on both Intel and PowerPC architectures. It is also a fully multi-threaded application: audio import and waveform caching are performed on separate background threads so that users can begin working immediately after selecting a file.

Evaluation

We recruited seven users (six students and one professional) to try out *Beat Tapper* and obtain qualitative feedback on DiMaß. We tested with both music, using a Vienna Philharmonic recording of *Blue Danube Waltz* by Johann Strauss, and speech, using an excerpt from an audio book of *Ein Wintermärchen* by Heinrich Heine. Users were encouraged to experiment with the viscosity setting, and with constant and variable pitch time-stretching.

Users readily grasped the directness of grabbing on to the waveform and shuffling it around. One user commented on how he expected the waveform to move with some "inertia", and thus continue to move even after he released the waveform. As expected, all users vastly preferred constant pitch time-stretching to variable pitch, where the audio resembles "a broken record"; such audio also becomes disturbing with rapid movements due to the increased pitch. Most users preferred a higher viscosity setting, between 0.5 and 0.9, as it produced smoother and more pleasant-sounding audio.

Conclusions

We presented DiMaß, a direct manipulation technique for interacting with the timeline of digital audio. DiMaß consists of a motion estimator, input tracker, and audio time-stretcher. We introduced an improved algorithm to synchronize audio to user input; this algorithm includes a "viscosity" setting that smoothens the adjusted play rate, but with a small impact on responsiveness. Informal user tests showed that even though our algorithm is capable of almost zero-latency synchronization at a low viscosity setting, users preferred a higher setting as it produced more pleasant-sounding audio. We hope our work on DiMaß will make scrubbing and searching through audio for both content producers and consumers a more efficient and pleasant experience.

The authors thank Thorsten Karrer for his work on the time-stretching algorithm and Henning Kiel his code contributions to *Beat Tapper*.

References

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

[2] H. Honing. From Time to Time: The Representation of Timing and Tempo. *Computer Music Journal*, 25(3):50–61, 2001.

[3] W. Hürst, T. Lauer, C. Bürfent, and Götz. Forward and Backward Speech Skimming with the Elastic Audio Slider. *Proceedings of the 19th British HCI Group Annual Conference*, Edinburgh, Scotland, 2005.

[4] T. Karrer, E. Lee, and J. Borchers. PhaVoRIT: A Phase Vocoder for Real-Time Interactive Time-Stretching. *Proceedings of the ICMC 2006 International Computer Music Conference*, New Orleans, USA, 2006. In Print.

[5] H. Kitamura. New Algorithms and Techniques for Well-Synchronized Audio and Video Streams Communcations. *Proceedings of the 6th International Conference on Computer Communications & Networks*, pp. 214–219, 1997.

[6] E. Lee. Drifting Into Sync. *Audio Anecdotes III: Tools, Tips and Techniques for Digital Audio*. Ed. K. Greenebaum and R. Barzel. AK Peters, 2006. In Print.

[7] E. Lee, T. Karrer, and J. Borchers. Toward a Framework for Interactive Systems to Conduct Digital Audio and Video Streams. *Computer Music Journal*, 30(1):21–36, 2006.

[8] B. Shneiderman. *Designing the User Interface*. Addision Wesley, 3rd edition, 1997.