Audio Engineering Society

Convention Paper

Presented at the 125th Convention
2008 October 2–5 San Francisco, CA, USA

SoundTorch: Quick Browsing in Large Audio Collections

Sebastian Heise[1], Michael Hlatky[1], and Jörn Loviscach[1]

[1] *Hochschule Bremen (University of Applied Sciences), 28199 Bremen, Germany*

Correspondence should be addressed to Jörn Loviscach (joern.loviscach@hs-bremen.de)

ABSTRACT

Musicians, sound engineers, and foley artists face the challenge of finding appropriate sounds in vast collections containing thousands of audio files. Imprecise naming and tagging forces users to review dozens of files in order to pick the right sound. Acoustic matching is not necessarily helpful here as it needs a sound exemplar to match with and may miss relevant files. Hence, we propose to combine acoustic content analysis with accelerated auditioning: Audio files are automatically arranged in 2D by psychoacoustic similarity. A user can shine a virtual flashlight onto this representation; all sounds in the light cone are played back simultaneously, their position indicated through surround sound. User tests show that this method can leverage the human brain's capability to single out sounds from a spatial mixture and enhance browsing in large collections of audio content.

1. INTRODUCTION

For at least the last two decades, the processes for managing, browsing and auditioning sound samples on a computer have remained the same. Auditioning mostly happens file by file, which consumes time and kills creativity. Sounds are not easy to compare because only one file is played back at a time, and files from different folders cannot be triggered in immediate sequence.

In theory, sound searches can be sped up if the audio files possess meaningful names or are equipped with tags. However, names and tags are hard to create and maintain in a consistent state if the number of audio files ranges in the thousands [7]. In addition, names and tags are not of great help when the perfect crackling noise for a campfire stems from an icebreaker ship. Content-based methods of Music Information Retrieval (MIR) may come to the rescue, in particular when combined with visualiza-

tion [4]. MIR helps to classify sounds by similarity. Currently, however, its reliability is not perfect so that one has to individually audition dozens of files and may still miss an interesting candidate. We propose a solution to this problem through combining MIR methods with a novel user interface for quick auditioning.

Our prototype represents audio files on a 2D screen by dots on a dark background, see Figure 1. The layout in 2D is based on methods from MIR, so that acoustically similar sounds are placed next to each other. To this end, Mel-Frequency Cepstral Coefficients (MFCCs) [6] are computed for every audio file; the files' positions are laid out through a Self-Organizing Map (SOM) [8]. This approach is proven to produce meaningful arrangements of songs concerning their genre [9, 10].

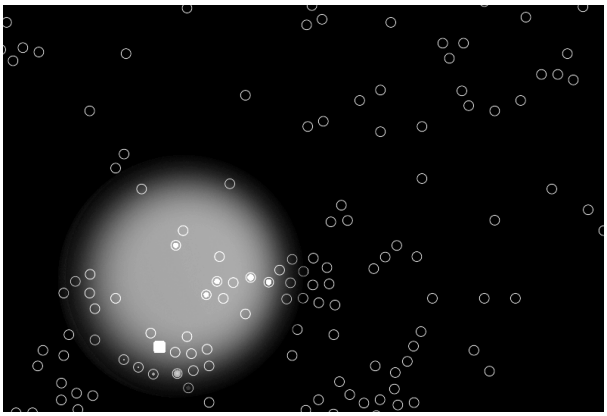

Fig. 1: The user can shine a virtual flashlight onto a collection of sound files.

The proposed quick auditioning method leverages the "Cocktail-Party Effect," the human capability for complex auditory scene analysis [2, 3]. Our system places the listener at the center of a graphical flashlight that illuminates the files to be auditioned. The audio of the illuminated files is mapped spatially to a user-defined loudspeaker setup. A computer mouse or a Nintendo Wii Remote steers the light cone emerging from the virtual torch over the 2D file representation. The illumination triggers the simultaneous playback of all sounds it highlights. The playback of every file repeats until the light beam is moved out of range. The auditory separation of the sounds is based on the mapping to a multi-channel loudspeaker system and is supported further by the fast interaction: The sound mix changes when moving the torch, so it is easier to locate single streams.

The cone's diameter and the zoom factor of the display are easily adjustable. This allows quick transitions between an overview of thousands of files and a close-up of dozens. There is no hard switching between files under audition but rather a continuous, glitch-free transition from inaudible to fully audible as the flashlight is moved around. The dots that represent the currently active files within the spotlight of the torch pulsate to act like VU meters. The visual output is mostly handled using the programmable features of current graphics cards.

Methods of Music Information Retrieval have mostly been applied to complete songs, not to single sounds. However, the Freesound Project (`http://www.freesound.org/`) employs user-provided tagging; AllThatSounds.net (`http://www.allthatsounds.net/`) additionally makes use of acoustic feature analysis. None of both projects is concerned with advanced visualization.

In concurrent work, Streich and Ong [13] visualize music loops in a 2D arrangement according to their similarity. The—optionally synchronized—playback can be switched on and off per loop. Our user interface also resembles StockSynth (`http://www.dontcrack.com/freeware/downloads.php/id/2279/software/StockSynth/`), a SuperCollider patch, which is intended for real-time performances and neither offers automatic layout nor more than stereo sound output. In Schmandt's "Audio Hallway" [12], short news samples from a radio network were grouped together in virtual rooms based on metadata. The user could navigate through a hallway hearing short clippings from the clusters and could than decide to enter a room. An additional text stream was used to do the grouping by text analysis. Dachselt and Frisch [5] dealt with the problem of visualizing a music collection by presenting different zoom levels. With its prominent use of zooming, SoundTorch, too, can be considered a specific case of a Zoomable User Interface [1].

This paper is structured as follows: Section 2 presents the methods used to lay out the sound files in a two-dimensional field. Section 3 covers the level

control employed to fade the files' content in and out as well as the panning method. The graphical and tangible user interface is described in Section 4. The implementation and the technical performance of our prototype are detailed in Section 5. We report results of an evaluation with test users in Section 6 and conclude the paper with Section 7.

2. 2D LAYOUT

Mörchen [9] provides a detailed description of mapping music genres with a self-organizing map using high-dimensional feature vectors extracted from sound files. Not dealing with complete music files but with comparatively simple sound effects, we decided to deal with a limited set of acoustic features. In a preprocessing step we extract 13 MFCCs for every frame (46 ms, 50 % overlap) of every sound file in the training set, which encompasses around 3000 sounds. Regions of the audio files whose level is below an adjustable threshold are ignored. Since most sounds effects possess a relatively stable timbre, we elected to further condense this number, for which reason we determined the centroid of a sound's MFCC cloud, similar to [9]. The median for every sound is then taken to train the SOM with different distance functions. We compared Euclidean distance, Mahalanobis distance, and the scalar product of the normalized vectors.

To be able to compare different maps, we refrained from the typical random initialization of the SOM. Instead, we initialize it with smooth gradients. To this end, the 13 elements of the MFCC vector are divided into three groups: $0 \ldots 4$, $5 \ldots 8$, $9 \ldots 13$. For every cell $(x, y) \in [0, W] \times [0, H]$ in the SOM's lattice, the components of the first group are filled with $\cos(\pi x/W)$, those of the second group with $\cos(\pi x/W) \cos(\pi y/H)$, and those of the third group with $\cos(\pi y/H)$. This very roughly amounts to placing noisy and dull sounds at large x and small y; toward small x, the sound gets more tonal; toward large y, pitch goes up. Figure 2 shows the training result. As Figure 3 shows, the cosine distance function tends to work best.

3. FADING, BLENDING, AND PANNING

The amount of light that shines onto a sound object on the surface determines its volume in the mix. The corresponding "illumination" levels of the sound files are computed using an invisible rendering pass

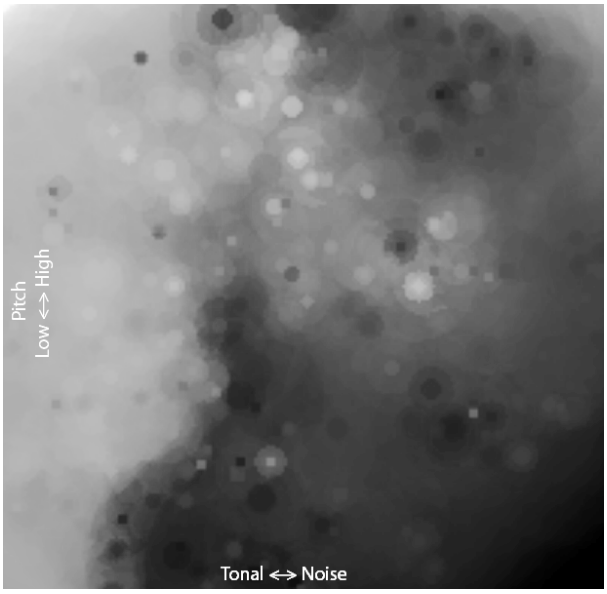

Fig. 2: The used SOM with feature vectors mapped to color.

on the graphics chip, which proved to be a major speedup in the computation. As long as a sound file is illuminated, it keeps playing back. If the flashlight moves out of range, the playback position for that specific sound is kept until it is illuminated again. During a preprocessing step, larger regions of silence can be eliminated and overly short sound files can be extended by silence, so as to create a more consistent sonic image.

When the light cone moves over the surface, every sound file is smoothly faded in and out. Audio levels of individual recordings may differ drastically and the number of files playing may range from a single one to several hundreds. Hence, strong dynamic compression is employed. Our multichannel environment is equipped for 2D spatialization. In the software prototype, the user can adjust the number and the positions of loudspeakers along the circumference of a circle to mimic the actual listening environment. This circle is virtually mounted to the light cone and moves with the cone's position, see Figure 4.

Having to deal with virtual sound sources that may appear in *front* of the loudspeakers, we cannot fully employ Pulkki's vector-based panning method [11].

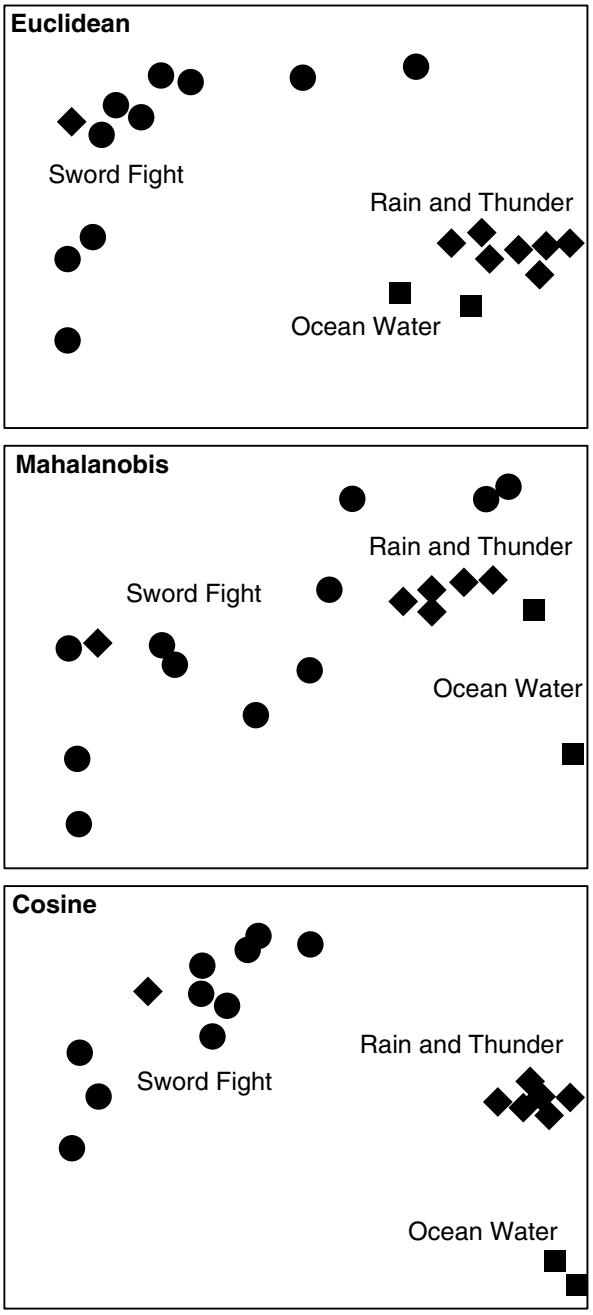

Fig. 3: Distribution of sound files using different distance functions. Note that one thunderstorm sound is misplaced in all three approaches.

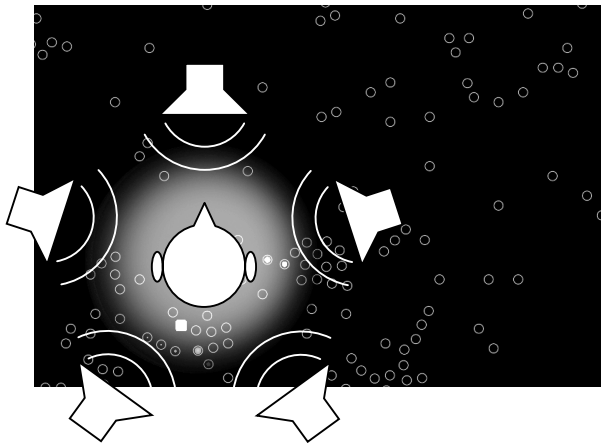

Fig. 4: The spatialization simulates that the user sits at the center of the light cone.

In particular, we have to avoid quick switches between loudspeaker pairs when a sound source gets close to the listener's position. Hence, we blend into a uniform distribution of the sound onto all loudspeakers, when this distance becomes small.

The level-based panning is enhanced by a delay that depends on the relative positions of the virtual loudspeakers and the sound source. The computation is based on the projection along the line that connects the listener's head with the sound source, see Figure 5. This projection yields a delay factor, which can be scaled according to the user's wishes. Since we are dealing with sound playback, the delay can be realized through offsetting the playback position, with no specific delay lines involved.

4. INTERACTION AND GRAPHICS

The position of the flashlight's cone can be controlled in three different ways: first, through the computer's mouse; second, through the keys W–A–S–D, like in a first-person shooter game; third, through the infrared sensor of the Nintendo Wii Remote. The latter also mimics the handling of a flashlight very closely; its four cursor keys can be used to invoke sideway motion.

There is no rotational motion so that the user always looks onto the surface from a right angle; in addition, the top of the computer screen is always mapped to the loudspeaker in front of the user. The zoom level

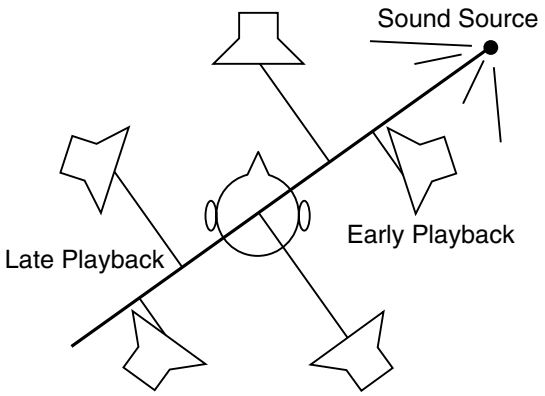

Fig. 5: The delay is computed with the help of the axial distance to the sound source.

can be adjusted with the scroll wheel of the mouse or by changing the distance between the Wii Remote and the screen.

If it is not used for zooming, one can apply the Wii Remote controller with one single beacon of infrared light, as only a 2D position needs to be determined. To measure forward and backward motion of the controller, at least two beacons are required. The field of view of the Wii Remote controller's infrared sensor is rather narrow, as it is intended to be used on a couch in front of a TV screen. Thus, this setup is not ideal for desktop computing. Another option for a user interface is to employ the Wii Remote's accelerometer to steer the position by tilting the controller; this is independent of additional beacons and hence works well on the desktop.

All sounds within the light cone are displayed as dots whose size and color pulsate according to the level. This visual representation enhances the integration of vision and sound: The pulsation, which resembles a VU meter, makes it more intuitive to link the dots on the screen to the spatialized audio signals than just the spatial distribution alone. If sounds outside the light cone would not be rendered visually, the user's orientation would suffer. Hence, they are drawn "ghosted": as thin rings of a fixed size.

5. IMPLEMENTATION

The prototype's implementation employs several data collections: first, a list with references to sound objects, the active—i. e. illuminated—ones being swapped to the beginning for maximum compactness; second, a list with $n \times 1$ mix matrices and delay offsets to control the distribution of every (mono) sound file onto n loudspeakers; third, a vertex buffer on the graphics card that stores the screen position and the current VU level for every sound object; fourth, a texture on the graphics card that contains a list of the coordinates of the sound objects on the SOM field, calculated on initialization through best matches with the SOM.

Even though the prototype works in 2D, it employs 3D coordinates internally. This requires no additional computational effort since the projection is done by the GPU, which is highly optimized for 3D operations. Due to the 3D coordinates it would be easy to switch to a 3D SOM mapping and treat the sound collection as a universe with clusters that appear as sound galaxies.

The process to compute the sound control data runs as follows, see Figure 6: First, the software checks if the input from the input control device (mouse, keyboard, Wii Remote controller) has changed. If not, the control data do not need to change and the next steps can be leaped over for this cycle. Then, the list of sounds is updated based on the currently stored illumination level of the sounds. Sounds whose state changed to active are swapped with the first inactive ones; sounds with zero output level are pushed to the list's tail by swapping them with the last active one. The mix matrix is updated for every active sound. Next, an invisible rendering step uses the texture containing the SOM coordinates of the sound objects to compute the screen positions of the sound objects and the amount of light they receive. This data is then used to update all points in the vertex buffer with new 2D coordinates. The sound card's mixing loop, which runs in a parallel thread, hands over level data that are fetched now and stored in the vertex buffer. Finally, the graphics card draws the new image frame.

The dots representing the sounds are rendered as squares sized 50×50 pixels, which are overlaid transparently. A pixel shader computes the distance of a pixel from the center. This distance is used to either render a ring (inactive state) or to produce a disk of adjustable size (active state) in the pixel shader. The remaining pixels are left transparent.

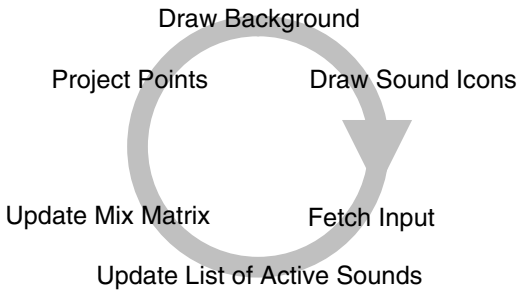

Fig. 6: The update process for the sound control data runs partially on the CPU and partially on the GPU.

Audio streams are read, decoded and buffered asynchronously ensuring that enough computing power remains for audio mixing and graphics calculation. On a standard notebook computer, the prototype can seamlessly play back 100 audio files in parallel. The number of visible files is virtually unlimited, since only the "illuminated" ones are processed. The audio output loop only needs to process the active sounds. The initial parts of all sound files on the surface are buffered in main memory to have all samples at hand without delay.

The prototype has been implemented in C# and HLSL on the .NET platform using Microsoft XNA Game Studio 2.0. Our .NET wrapper for audio card drivers conforming to Steinberg's ASIO interface ensures high-performance multichannel output. If no ASIO device is available, the prototype falls back to the Microsoft Windows Audio Session API (WASAPI), which still provides a short-latency playback. For decoding the different types of audio files we employed a proprietary .NET wrapper around the BASS audio library (`http://www.un4seen.com/`).

We tried out different collections of sound effects databases from `http://www.sound-ideas.com/`, in total around 5000 sound files, amounting to nine gigabytes of MP3 files. Tests were conducted on a standard 1.8 GHz Intel Centrino Duo notebook with an NVidia GeForce 7400 Go graphics and 2 GB main memory. This computer can handle the complete database while still keeping a speed of the drawing loop of about 60 Hz. The pre-buffered audio data consume about 650 MB of memory. The loudspeaker setup employed in the tests was a regular arrangement of five near-field studio monitors.

As described before, only the active sounds are processed, which helps to keep the system load manageable. It is reasonable to limit the number of active sound files to a 100 or even less because distinguishing sounds in such a mixture becomes impossible. During testing, we found that one performance problem could arise due to the number of opened files. Opening several thousand files during the initialization phase places a huge burden on the operating system. It may be advisable to first create a single combined file containing all audio samples in sequence.

6. EVALUATION

We conducted a think-aloud user test with 15 subjects (6 female, 9 male; age 23 to 43, median 27), all of whom are computer-literate or sound professionals. Every user had to tackle three problems both with SoundTorch and with the list-based auditioning function of Sony Vegas 8. We confronted the subjects with different sound sets of 100 files each, taken from Sound Ideas's "The General" Series 6000 (`http://sound-ideas.com/sfxmenu-6000.html`), which covers a broad variety of sound effects for movie production. A different set of sounds was selected for every experiment; every user worked with the same set for that specific experiment. The entries of the lists, however, were shuffled randomly for every user. The file names where changed to numbers so that it was not possible to draw a conclusion from them.

6.1. Task 1: Gain an Overview

After an introduction to the interface of SoundTorch, the first task was to explore an unknown set of sound files and gain an overview of its content. We asked the participants to mentally form an appropriate grouping of the sounds. This task was mainly used to familiarize the users with the interface. Nonetheless, already in this test it turned out that the task was easier to solve with SoundTorch: 12 of the 15 subjects identified the (relatively obvious) classes of sounds; only one participant succeeded with the list-based interface.

6.2. Task 2: Search for a Given Sound

The second task was to find given sounds, namely a "blib" and a "ding dong" using SoundTorch and

the list-based auditioning tool. If a user forgot the pattern he or she had the possibility to listen to the pattern whenever needed. It turned out that this task was solved extremely quickly with SoundTorch, see Figure 7. The high largest value was due to one subject being confused by the mapping, which we consider a training issue. 14 of the 15 participants could solve this task with SoundTorch, only 8 participants succeeded with the list-based interface.

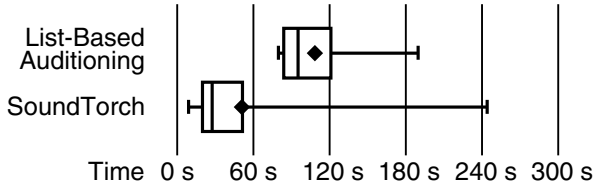

Fig. 7: SoundTorch drastically speeds up the search for a given sound.

6.3. Task 3: Search for an Unknown Sound

Here, the subjects had to search for audio samples that can be used to produce a sound that does not exist in reality. First, the participants were instructed to come up with ideas to produce a dinosaur sound from 100 samples that included animal sounds from pigs, sheep, and cows, but also some machine and nature sounds (total playing time: 130 minutes). Second, they had to find material for an alarm sound in an extraterrestrial spacecraft, for which we provided another set of 100 sounds: synthesized noises, servo motors, noises from sport events, and war sounds (total playing time: 70 minutes).

It turned out that with Sound Torch all users reported that they enjoyed exploring the set and began to weigh different sounds against each other. In particular, the subjects spend three minutes on average in browsing though the different sounds. All of them were satisfied with the result they found. With the list-based auditioning tool, most users claimed to be bored by "having to listen to all that stuff" and finished the task after one minute. Here, only 10 of the 15 subjects came up with a result that pleased them; this result was always the first one they encountered that fulfilled their needs.

6.4. Overall Observations

In our interviews with the candidates, it turned out that working with SoundTorch is much more pleasing than stepping through a list. However, users with no professional audio expertise reported that they felt overtaxed with the focused listening to many simultaneous sounds for several minutes. On top of that, the non-professionals hesitated to engage the full power of the system by using a large light cone to play back a dozen of sounds at the same time.

Further experimentation with the user interface has shown that attaching names to representational dots is counterproductive since the user tends to focus on reading the labels. In addition, labels or tags may be misleading rather than helpful.

7. CONCLUSION AND OUTLOOK

This work demonstrated a novel approach to explore sound databases. It produces results quicker than stepping through a linear list of files for auditioning. In addition, it entertains its users and thus promotes serendipity.

The system is based on surround sound, but may also be used with stereo sound reproduction. Even though the fall-back solution for stereo systems is already promising, the use of headphones could be enhanced by using HRTFs for spatialization. This could mean to port more parts of the audio processing to the GPU. One can try to use a 3D mapping to fly through galaxies of sounds instead of walking through clusters of sounds on a surface. The underlying engines already provide the necessary functionality. An adjacency map for the sound icons could help limiting all updates to the corresponding neighborhood and thus enhance performance.

The mapping algorithm still presents issues: Currently our mapping only covers the use of MFCCs. Several other acoustic features such as those defined in the MPEG-7 standard can be used alone or in combination with our technique. In particular, the temporal development of a sound is currently ignored. Finally, one could optimize the distance function for the self-organizing map and the SoundTorch projection in a training phase.

Future work may also address new zooming interactions. For instance, in the spirit of "liquid browsing" [14] one could create rather a sound *lens* than a sound *torch*.

8. REFERENCES

[1] B. Bederson and J. Meyer. Implementing a zooming user interface: experience building Pad++. *Softw. Pract. Exper.*, 28(10):1101–1135, 1998.

[2] A. S. Bregman. *Auditory Scene Analysis*. MIT Press, Cambridge, Massachussetts, 1990.

[3] E. Cherry. Some experiments in the recognition of speech, with one and two ears. *J. Acoustical Soc. of America*, 25:975–979, 1953.

[4] M. Cooper, J. Foote, E. Pampalk, and G. Tzanetakis. Visualization in audio-based music information retrieval. *Comput. Music J.*, 30(2):42–62, 2006.

[5] R. Dachselt and M. Frisch. Mambo: a facet-based zoomable music browser. In *MUM '07: Proceedings of the International Conference on Mobile and Ubiquitous Multimedia*, pages 110–117, 2007.

[6] J. Foote. Content-based retrieval of music and audio. In C.-C. J. Kuo, editor, *Multimedia Storage and Archiving Systems II, Proceedings of SPIE*, pages 138–147, 1997.

[7] J. Foote. An overview of audio information retrieval. *Multimedia Systems*, 7(1):2–11, 1999.

[8] T. Kohonen. *Self-organizing maps*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1997.

[9] F. Mörchen. Modelling timbre distance with temporal statistics from polyphonic music. *IEEE Transactions on Speech and Audio Processing*, 14(1):81–90, 2006.

[10] E. Pampalk, A. Rauber, and D. Merkl. Content-based organization and visualization of music archives. In *MULTIMEDIA '02: Proceedings of ACM International Conference on Multimedia*, pages 570–579, 2002.

[11] V. Pulkki. Virtual sound source positioning using vector base amplitude panning. *J. Audio Eng. Soc.*, 45(6):456–466, 1997.

[12] C. Schmandt. Audio hallway: a virtual acoustic environment for browsing. In *UIST '98: Proceedings of the ACM Symposium on User Interface Software and Technology*, pages 163–170, 1998.

[13] S. Streich and B. S. Ong. A music loop explorer system. Appears in: ICMC '08: Proceedings of the International Computer Music Conference, 2008.

[14] C. Waldeck and D. Balfanz. Mobile liquid 2D scatter space (ML2DSS). In *IV '04: Proceedings of the Information Visualization Conference*, pages 494–498, 2004.