MARSYAS3D: A PROTOTYPE AUDIO BROWSER-EDITOR USING A LARGE SCALE IMMERSIVE VISUAL AND AUDIO DISPLAY

George Tzanetakis

Computer Science Dept.
Princeton University
NJ 08544 USA
`gtzan@cs.princeton.edu`

Perry Cook

Computer Science and Music Dept.
Princeton University
NJ 08544 USA
`prc@cs.princeton.edu`

ABSTRACT

Most audio editing tools offer limited capabilities for browsing and editing large collections of files. Moreover working with many audio files tends to clutter the limited screen space of a desktop monitor. In this paper we describe MARSYAS3D, a prototype audio browser and editor for large audio collections. A variety of 2D and 3D graphics interfaces for working with collections and/or individual files have been developed. Many of these interfaces are informed by automatic content-based audio analysis tools. Although MARSYAS3D can be used with a desktop monitor it has been specifically designed as an application for the Princeton Scalable Display Wall project.

The current Display Wall system has an 8 x 18-foot rear projection screen with a resolution of 4096 x 1536 pixels. For sound a custom-made 16-speaker surround system is used. The users interact with the Wall using a variety of input methods. This immersive display allows for visual and aural presentation of detailed information for browsing and editing large audio collections and supports natural interactive collaborations among multiple simultaneous users.

1. INTRODUCTION

Continuous rapid-improvements in CPU performance and storage density have made possible the collection of large amounts of audio data. The ability to interchange files over the Web and audio compression formats like MP3 have further facilitated this trend. Most audio editing tools offer limited capabilities for browsing and editing large collections of audio files (typically only the filename and some form of ID tagging is used).

In this paper we describe MARSYAS3D, a prototype audio browser and editor designed for working with large collections of audio files. A variety of 2D and 3D graphical interfaces for working with collections and/or individual files have been developed. Many of these interfaces are informed by automatic content-based audio analysis tools. These tools are all integrated under a common underlying framework that provides a unique consistent data representation.

Working with many audio files using a desktop monitor is difficult because of the large number of necessary interfaces and windows. Although MARSYAS3D can be used with a desktop monitor it has been specifically designed as an application for the Princeton Scalable Display Wall project. This large immersive display allows for visual and aural presentation of detailed information for browsing and editing large audio collections and supports interactive collaborations among multiple simultaneous users.

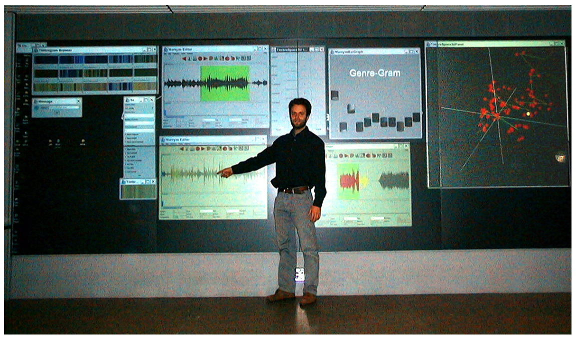


Figure 1: MARSYAS3D on the Display Wall

1.1. The Wall

The Princeton Scalable Display Wall project explores building and using a large-format display with multiple commodity components. The prototype system has been operational since March 1998. It comprises an 8 x 18 foot rear-projection screen with a 4 x 2 array of projectors. Each projector is driven by a commodity PC with an off-the-shelf graphics accelerator. The multi-speaker display system has 16 speakers around the area in front of the wall and is controlled by a *sound server* PC that uses two 8-channel sound cards. A more detailed description of the project can be found [1]. Figure 1 shows MARSYAS3D on the Princeton Display Wall. High resolution color images can be found at

http://www.cs.princeton.edu/˜gtzan/marsyas.html.

1.2. Applications

The driving application behind the development of MARSYAS3D is research in content-based audio information tools. Other potential applications are auditory display research, sound effects for movies, music browsing, computer music and in general any application area where there is a need to work on large collections of digital audio.

We loosely define as *large* any collection with more than 100 audio files. Empirically this is the approximate number of files where working only with filenames and ID tags becomes tedious. Collections can have an arbitrary number of files subject to the memory and disk constraints of the particular system. The client-server and modular architecture of the system makes it scalable to more than one computer. In addition it facilitates interaction with non-standard input devices like Speech Recognition.

As a typical application scenario imagine having to create the soundscape of walking through a forest using a database of sound effects. As a first step one might look for the sound of walking on soil. After clustering the collection you would notice that a particular cluster has many files with Walking in their filename. Expanding on this cluster you would select 4 sound files labelled as walking on soil. Using spatialized audio you could automatically hear generated thumbnails of the 4 sound files simultaneously and make the final decision. In a similar fashion you could locate some files with bird calls and a file with the sound of a waterfall. Once the components of the soundscape are selected you can edit the selected soundfiles and perform the necessary mixing operations to create the final soundscape of walking through a forest. The illusion of approaching the waterfall can be achieved by artificial reverberation and crossfading.

All these tasks are performed in front of a large screen with multiple graphics displays that are all consistent and connected to the same underlying data representation module. Of course having a large collaborative space allows more than one user to interact simultaneously with the system.

2. RELATED WORK

A significant influence for this project has been the music browsing work of [2, 3]. In this work, a prototype 2D graphics browser combined with multi-stream sonic browsing was developed and evaluated. The driving application behind the development of their browser was musicological research in Irish traditional music although their browser could easily be applied to other applications. In addition to offering similar browsing capabilities our system is assisted by automatic content-based audio analysis tools and supports 3D graphics displays. Some of the 3D graphics tools have been described in [4]. Furthermore, once the desired files have been located, sound editing capabilities are provided.

Commercially available audio editors are either small scale editors that work with relatively short sound files, or professional tools for audio recording and production. In both cases they offer very little support for browsing large collections. Typically they support montage-like operations like copying, mixing, cutting and splicing and some signal processing tools such as filtering, reverberation and flanging. In addition to these standard functionalities, our editor offers content-based audio analysis tools like segmentation and timbre-based coloring to assist the editing process.

Although there have been attempts [5, 6] to develop “intelligent” sound editors most of their research effort was spend addressing problems related to the limited hardware of that time. Audio analysis tools have been used succesfully more recently in [7], where speech recognition techniques are employed to index audio and video.

3. SYSTEM ARCHITECTURE

The system architecture has been designed with the Shneiderman mantra for user interfaces **”overview first, zoom and filter, then details on demand”** in mind [8]. Moreover the user can configure the system to suit his specific needs and it is easy for the programmer to extend the system with new interfaces. A very important characteristic of the system is that although there are multiple interfaces and views everything is connected and updated synchronously from the same underlying data representation.

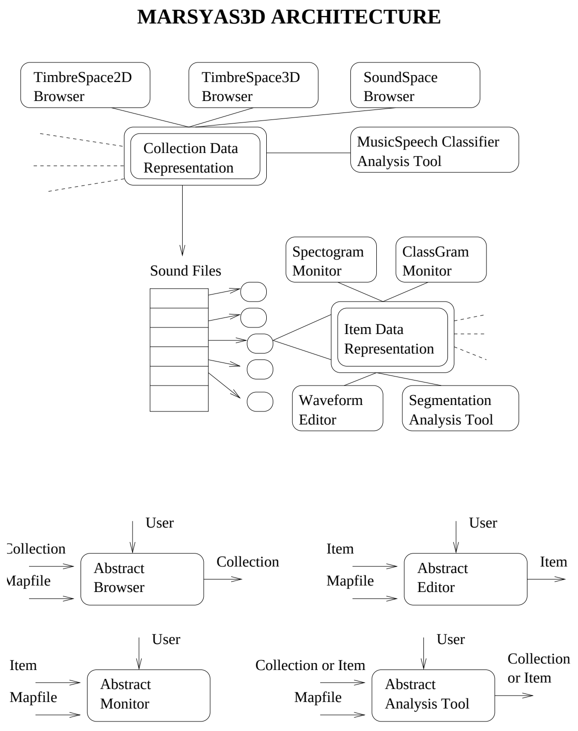


Figure 2: *MARSYAS3D System Architecture*

Central to the design of the system is the concept of browsing. Marchionini and Shneiderman [9] define browsing as:

- an exploratory, information seeking strategy that depends upon serendipity
- especially appropriate for ill-defined problems and for exploring new task domains

Working with large audio collections has those characteristics, therefore a variety of different *Browsers* are provided. The *Browsers* render the collection based on parameters stored in *MapFiles*. In many cases the *MapFile* can be automatically generated by automatic *Audio Analysis Tools*. Using the *Browser* the user can explore the space and select a subset of the collection for subsequent browsing or editing. Editing the selected files can be done using the *Editor* which is also configured by a *MapFiles*, possibly automatically generated. During playback of the files various *Monitors* can be setup to visual information that is updated in real-time. A schematic of the system’s architecture can be seen in Figure 2.

3.1. Terminology

The following terms will be used for the more detailed description of the system’s components:

Collection is a list of audio files. There is a unique underlying data representation for collections that is shared among all the *Browsers*. Any subset of a collection is also a collection.

Browsers present the collection for browsing using different audio and visual displays. The user can explore the space and select specific items for subsequent browsing or editing.

MapFiles map the individual items of the collection to the parameters required by a specific *Browser, Editor* or *Monitor*.

MapFileBuilder : is a tool for creating *MapFiles* either automatically using *Audio Analysis Tools* or manually.

Analysis Tools : apply signal-processing feature extraction and pattern recognition techniques to automatically infer information about the content of collections and individual items. In some cases applying an analysis tool to a collection corresponds to batch processing of all the individual files.

RealTime monitors : are 2D or 3D graphic displays that are updated in real-time while the sound is playing.

Item Editor : is the only component that affects the actual content of the files. It offers automatic and semi-automatic audio analysis tools in addition to standard editing and digital audio effects.

In Figure 1 the following interfaces are shown on the display wall: three *Editors*, a ClassGram *Monitor*, a TimbreGram *Browser*, and a TimbreSpace3D *Browser*.

4. AUDIO ANALYSIS TOOLS

Audio analysis tools apply signal processing and pattern recognition techniques to extract information about the content of audio files. Examples of audio analysis tools are classification, segmentation, similarity-retrieval, thumbnailing, principal component analysis (PCA), beat-detection and clustering.

Examples of classifiers supported by the system are: Music vs Speech, Male vs Female vs Sports, Music Genres and Instrument families. They have been statistically trained from representative datasets and evaluated [10]. Segmentation refers to the process of detecting segments when there is a change of “texture” in a sound stream. The chorus of a song, the entrance of a guitar solo or a change of speaker are examples of segmentation boundaries. Segmentation can be used to break a single soundfile into segments that can be subsequently viewed as a collection using the “Explode” operation.

Audio thumbnailing is the process of creating a short summary sound file that best captures the essential elements of a large sound file. Thumbailing is very important for quick browsing using the *SoundSpace* browser. Clustering can be used to hierarchically group files where no labelling information is provided. Beat-detection algorithms [11] measure the rhythmic beats-per-minute (bpm) for music and. Finally PCA is a dimensionality-reduction technique [12], used in our system to map high-dimensional spectral feature spaces to lower-dimensional MapFile spaces. More details about these techniques can be found in [13, 10].

An important aspect of MARSYAS3D is that all these techniques are integrated under a common framework and are used to support the *Browsers*, *Editor*, and *Monitors*. For example in a *TimbreSpace*, PCA can be used to map audio files to points in 3D space and clustering can be used to color them. Segmentation can be used to quickly locate a region for editing and classification can be used to inform a *ClassGram* monitor. The flexibility provided by *MapFiles* does not enforce specific techniques for each type of display. For example a *TimbreSpace2D* could easily be constructed by manually moving points in a 2D plane as well as with automatic techniques.

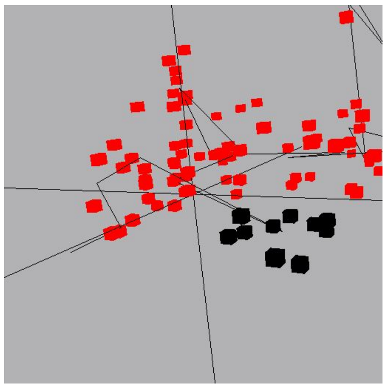


Figure 3: *TimbreSpace3D*

5. BROWSERS

The key operation any browser must support is “Zooming”. By zooming a selected subset of a collection is opened as a new collection. This is easy to implement because any subset of a collection is also a collection (same for the corresponding MapFile). Each browser also supports a currently selected item. It is important to note that since all browser are updated synchronously any possible combination of them can be used.

The following browsers are implemented in the system:

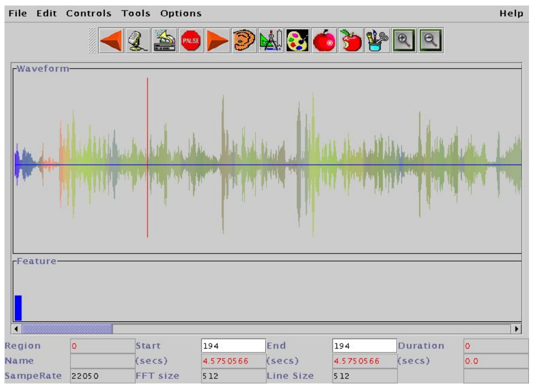
Standard is just a list of fileanames corresponding to the audio files. Standard single and multiple mouse selection is supported.

Tree renders the items based on a tree representation. This tree representation can be generated by automatic classification. For example the first level of the tree could be Music vs Speech and Speech could be further subdivided to Male vs Female and so forth.

TimbregramBrowser renders each file as *TimbreGram* that are rectangles consisting of vertical color stripes where each stripe corresponds to a short segment of sound. Time is mapped from left to right. They reveal sounds that are similar by color similarity and time periodicity. For example the *TimbreGrams* of two walking sounds with different number of footsteps will have the same periodicity in color. Different color mappings can be used and PCA can be applied to reduce spectral features to the color parameters for each stripe. Figure 4 shows *TimbreGram* coloring superimposed over a waveform display.

TimbreSpace2D represent each item as a geometric shape on plane in a similar fashion to the visual browser of [2, 3]. The placement, shape and color of each item are controlled by the *Mapfile.* PCA can be used for automatic placement and classification or clustering can be used for automatic coloring or shape.

TimbreSpace3D is similar to the *TimbreSpace3D* in 3 dimensions. The user can zoom, rotate and translate the space. Principal curves [14] can be used to move sequentially through the data. This display is especially appropriate for the display wall where 3D illusion is much stronger. Figure 3 shows a *TimbreSpace3D* browser.



Region	0	Start	194	End	194	Duration	0
Name		(secs)	4.5750566	(secs)	4.5750566	(secs)	0.0
SampeRate	22050	FFT size	512	Line Size	512		

Figure 4: *MARSYAS3D Editor*

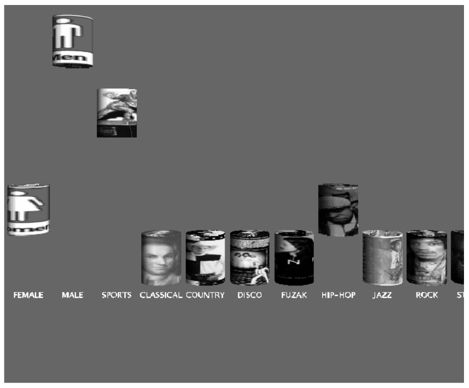


Figure 5: *ClassGram Browser (music-speech genres*

SoundSpace takes advantage of the cocktail-party effect in a similar fashion to the music browser of [2, 3]. As the user browses, a neighborhood (aura) around the currently selected item is auralized by playing all the corresponding files simultaneously. In the *MapFile* each file is mapped to one of the 16 loudspeakers of the system and the intensity levels are normalized for better presentation. Although all 16 loudspeakers can be used we have found that at most 8 simultaneous audio streams are useful. Using more advanced 3D audio rendering techniques to place files anywhere in the space is planned for the future. To shorten the playback time, equal length thumnails can be automatically generated for each file.

6. EDITOR

The editor supports standard operations like cutting, copying, pasting, splicing etc. Those operations work in the individual file and between different files that are open at the same time. A limited number of digital audio effects like reverberation and filtering are supported. Adding new effects to the system is easy.

In addition to these standard editor features Audio analysis tools like classification, segmentation, thumbnailing can assist the editing process. Computer-assisted annotation is also provided. As an example, a user can quickly detect the saxophone solo in a Jazz piece and annotate it with a textual description. Another feature of the editor is "Explode" which uses automatic segmentation to break the audio and then creates a collection of the segments for browsing. Figure 4 show the MARSYAS3D Editor with a waveform display with *TimbreGram* coloring.

7. MONITORS

Monitors display information that is updated in real-time while the audio files are being played. In many cases, they use the results of automatic audio analysis tools that work in real-time.

Waveform is the traditional display of the signal used by commercial editors.

Spectrum is a traditional spectrum display showing the magnitude in Decibels of the Short Time Fourrier Transform (STFT) of the signal.

Spectogram renders the same information as the Spectrum as a greyscale image. Time is mapped from left to right and the frequency amplitudes are mapped to greyscale values according to their magnitude.

Waterfall shows a waterfall-like 3D display of a cascade of Spectrum plots in 3D space.

TimbreBall shows a sphere moving a 3D cube. Each axis of the cube can be mapped to a specific audio analyss feature. For example, loudness, spectral centroid, and spectral flux (features known to be perceptually important) could be mapped to the x,y,z axis. A shadow is provided for better depth perception.

Amplitude shows an amplitude meter for the signal.

Metronome shows the bpm of the signal and a confidence level for how strong the beat is. The beat is detected automatically using techniques similar to [11].

FeaturePlot shows a plot of a user-selected feature.

ClassGram shows the relative weights of classification confidence. For each class a confidence measure, ranging from 0.0 to 1.0 is calculated and used to move up or down cylinders corresponding to each class. The cylinders can be texture-mapped with characteristic images for each class. This display can reveal detailed information about the underlying content. For example in a rap song both the "Male Speech" cylinder and the "Rap" cylinder would go up.

8. MAPFILES

MapFiles control the appearance and behaviors of the various interfaces in MARSYAS3D by mapping the filenames to specific parameters. For example in a *TimbreSpace3D* the filenames would be mapped to 3D position, shape and color. These parameters could be automatically generated by PCA, classification and clustering or could be created manually. In a *SoundSpace* each file would be mapped to a specific loudspeaker and intensity. *MapFileBuilder* is the application for creating*MapFiles* for collections. Currently it is a separate application without a graphical user interface.

9. COLLECTIONS

A variety of large audio collections were used for developing and testing MARSYAS3D. The length of the audio files varies from 30 seconds (most of the collections) to 3-4 minutes. The following collections where utilized:

MusicSpeech 200 audio clips speech and music

Radio 100 audio clips recorded from FM radio

Jazz 100 jazz audio clips recorded from CDs

Instruments 913 isolated musical instrument notes from the McGill MUMS CDs

Sfx 200 sound effects clips recorded from various sound effect libraries

10. IMPLEMENTATION

The software is implemented following a client-server architecture. All the computation-intensive signal processing and pattern recognition algorithms required for audio analysis are performed using a server written in C++. The code is optimized resulting in real-time feature calculation and monitor updates. The collection browser, item editor, real-time monitors and the integration code are written in JAVA and JAVA3D. The software runs on Linux, Solaris, Irix and Windows (95,98,00,NT). A free software version of some of the components is provided in :

http://www.cs.princeton.edu/˜gtzan/marsyas.html

The same link contains color pictures of the figures and additional information related to this paper.

11. FUTURE DIRECTIONS

The system is still a research prototype under development so there has been no formal evaluation. A task-based evaluation is planned for the future. In this evaluation, comparison between the effectiveness of different browsers and their combinations will be made. In addition the system will be compared to more traditional forms of browsing and editing.

The Display Wall project has implemented a variety of alternative input methods in addition to standard keyboard and mouse navigation. We are exploring the use of speech recognition input for controlling the system. The client architecture of the user interfaces will make the integration with the speech system easy. Speech recognition is already being used for a circuit viewer. Currently the placing of the windows is done by the window manager requiring manual adjustment which is difficult on such a large display. Therefore, intelligent automatic placement of windows is planned for the future.

Currently all the book keeping (mainly the *MapFiles*) is done using flat text files. We are considering the use of a database system to organize this information. Finally a graphical user interface for the *MapFileBuilder* is under development.

Our hope is that someday in the future, sound editing will look more similar to our prototype system than to what is currently commercially available.

12. ACKNOWLEDGEMENTS

Thanks to Doug Turnbull for work on the ClassGram for FM Radio and Jazz. This work was funded under NSF grant 9984087 and from gifts from Intel and Arial Foundation.

13. REFERENCES

[1] Kai Li et.al, “Building and using a scalable display wall system,” *IEEE Computer Graphics and Applications*, vol. 21, no. 3, July/August 2000.

[2] M. Fernstorm and C. McNamara, “After direct manipulation - direct sonification,” in *Proc.Int.Conf on Auditory Display, ICAD*, 1998.

[3] M. Fernstorm and L. Bannon, “Multimedia browsing,” in *Proc.Int.Conf on Computer Human Interaction, CHI*, 1997.

[4] G. Tzanetakis and P. Cook, “3D graphics tools for sound collections,” in *Proc. COSTG6 Conference on Digital Audio Effects, DAFX*, 2000.

[5] C. Chafe, B. Mont-Reynaud, and L. Rush, “Toward an intelligent editor of digital audio: Recognition of musical constructs,” *Computer Music Journal*, vol. 6, no. 1, pp. 30–41, 1982.

[6] J.A. Moorer, “The lucasfilm audio signal processor,” *Computer Music Journal*, vol. 6, no. 3, pp. 30–41, 1982, also appears in ICASSP 82.

[7] A. Hauptmann and M. Witbrock, “Informedia: News-on-demand multimedia information acquisition and retrieval,” in *Intelligent Multimedia Information Retrieval*, chapter 10, pp. 215–240. MIT Press, Cambridge, Mass., 1997, http://www.cs.cmu.edu/afs/cs/user/alex/www/.

[8] B. Shneiderman, *Designing the user interface: strategies for effective human-computer interaction*, Addison-Wesley, 1987.

[9] G. Marchionini and B. Shneiderman, “Finding facts versus browsing knowledge in hypertext systems,” *IEEE Computer*, vol. 19, pp. 70–80, 1988.

[10] G. Tzanetakis and P. Cook, “Audio information retrieval (AIR) tools,” in *Proc. Int. Symposium on Audio Information Retrieval, ISMIR*, 2000.

[11] E. Scheirer, “Tempo and beat analysis of acoustic musical signals,” *J.Acoust.Soc.Am*, vol. 103, no. 1, pp. 588,601, Jan 1998.

[12] L.T.Jolliffe, *Principal Component Analysis*, Springer-Verlag, 1986.

[13] J. Foote, “An overview of audio information retrieval,” *ACM Multimedia Systems*, vol. 7, pp. 2–10, 1999.

[14] T. Hermann, P. Meinicke, and H. Ritter, “Principal curve sonification,” in *Proc. Int. Conf on Auditory Display, ICAD*, 2000.