A framework for audio analysis based on classification and temporal segmentation

George Tzanetakis [1]
Department of Computer Science
Princeton University

Perry Cook [2]
Department of Computer Science [3]
and Department of Music
Princeton University

Topic areas: music interfaces, content-based retrieval, audio segmentation, audio browsing, machine listening

Neither this paper nor any version close to it has been or is being offered elsewhere for publication. All necessary clearances have been obtained for the publication of this paper. If accepted, the paper will be made available in camera-ready forms by June 14th, 1999, and it will be personally presented at the EUROMICRO 99 Conference by the author or one of the co-authors. The presenting author(s) will pre-register for EUROMICRO 99 before the due date of the camera-ready paper.

[1] gtzan@cs.princeton.edu
[2] prc@cs.princeton.edu
[3] Address: 35 Olden Street,Princeton NJ 08544 Fax: 609-258-1771

Abstract

Existing audio tools handle the increasing amount of computer audio data inadequately. The typical tape-recorder paradigm for audio interfaces is inflexible and time consuming, especially for large data sets. On the other hand, completely automatic audio analysis and annotation is impossible using current techniques.

Alternative solutions are semi-automatic user interfaces that let users interact with sound in flexible ways based on content. This approach offers significant advantages over manual browsing, annotation and retrieval. Furthermore, it can be implemented using existing techniques for audio content analysis in restricted domains.

This paper describes a framework for experimenting, evaluating and integrating such techniques. As a test for the architecture, some recently proposed techniques have been implemented and tested. In addition, a new method for temporal segmentation based on audio texture is described. This method is combined with audio analysis techniques and used for hierarchical browsing, classification and annotation of audio files.

1 Introduction

There is a growing amount of audio data available on the Internet and elsewhere today. The traditional tape-recorder sample-playback paradigm for browsing, locating, manipulating and skimming audio is cumbersome and inflexible. The main reason is that it treats audio data as a linear block of samples. Traditional information retrieval (IR) [22], used by many of the popular Web search engines, uses computer-readable text as data and offers the ability to quickly locate and browse large amounts of data using the familiar search and "ranked by similarity" interface. Unfortunately, there are no equivalent methods available for audio.

An obvious solution to the problem of handling large amounts of audio data is to annotate it with textual information and then use traditional IR techniques for searching. This approach works well and has the advantage of using well-known and supported techniques. On the other hand, using current interfaces human annotation of audio is extremely time-consuming.

Recently, a number of techniques for automatic analysis of audio information have been proposed [6]. These approaches work reasonably well for restricted classes of audio. Based on these techniques, a completely automatic annotation system for audio could be envisioned. Although not impossible in theory, there are two problems with such an approach. The first is that current systems are not perfect and, therefore, annotation errors are inevitable. This problem has to do with the current state of the art, so it is possible that in the future it will be solved. There is a second problem, however, that is more subtle and not so easy to address. Audio, and especially music, is heard and described differently by each listener. There are, however, attributes of audio that most listeners will agree upon, like the general structure of the piece, the style, etc. Ideally a system for annotation should automatically extract as much information as it can and then let the user edit and expand it.

This leads to a semi-automatic approach that combines both manual and fully-automatic annotation into a flexible, practical user interface for audio manipulation. This paper describes a framework for building audio analysis tools and integrating them using a semi-automatic graphical interface. The framework has been designed to be flexible and to accommodate new algorithms easily.

In addition, a new approach to segmentation of audio files based on texture is described. The combination of temporal segmentation and sound classification significantly reduces the overhead of manual annotation and forms a powerful foundation for audio analysis applications. Moreover, it can be used to improve classification performance by using a texture-adaptive window size for integrating classification results.

1.1 Applications

There are several possible application areas for semi-automatic audio analysis tools. Digital video libraries are an active area of research that could benefit from the development of such tools. The Informedia project at Carnegie Mellon [8] contains a terabyte of data. Indexing the archive is done using a combination of speech-recognition, image analysis and keyword searching techniques. Audio analysis and browsing tools would enhance the current indexing techniques, especially for the regions that do not contain speech.

Detecting speech segments is very important for automatic speech recognition systems especially when dealing with real world multimedia data. Moreover, detecting if a speaker is male or female or determining his identity improves recognition performance.

More generally, audio analysis tools can be used to implement signal responsive algorithms. A compression algorithm, for example, knowing that a signal contains speech can use a method that takes advantage of speech characteristics to achieve a higher compression rate. Responding to parameters of analyzed audio can also be used by interactive algorithms performing animation or sound synthesis in virtual reality simulations and computer games.

There are many libraries of sound effects and instrument samples available. Due to their large size, searching for a particular sound can be a daunting task. Using audio-similarity retrieval techniques and fast browsing can greatly accelerate this process.

1.2 Related Work

A number of techniques for audio analysis have recently been proposed. In this section, some of these systems , relevant to our work, will be briefly described. A more complete overview can be found in [6].

A robust multi-feature music/speech discriminator is described in [18]. A similar discriminator is used in [15] to initially separate speech from music and then detect phonemes or notes accordingly. A multi-feature classifier based on spectral moments for recognition of steady state instrument tones is described in [7].

A retrieval-by-similarity system for isolated sounds has been developed at Muscle Fish LLC [23]. Users can search for and retrieve sounds by perceptual and acoustical features, can specify classes based on these features and can ask the engine to retrieve similar or dissimilar sounds.

Speech Skimmer [1] is an example of pushing audio interaction beyond the tape-recorder metaphor. The user can audition spoken documents at several times real-time, using time compression techniques and segmentation based on pitch. Hidden Markov Models are used in [2, 10] for segmentation and analysis of recorded meetings by speaker.

2 Framework

All these projects use similar features, classifications and algorithms for different tasks. Therefore, in the design of our system, we made an effort to abstract the common elements and use them as architectural building blocks. This facilitates the integration of different techniques under a common framework and interface. In addition, it helps rapid prototyping since the common elements are written once, and developing and evaluating a new technique or application requires writing only the new task-specific code.

Typically sound analysis systems follow a bottom-up processing architecture where sensory information flows from low-level signals to higher level cognitive representations. However, there is increased evidence that the human auditory system uses top-down as well as bottom-up information flow [19]. A top-down (prediction-driven) approach has been used for computational auditory scene analysis [5]. An extension to this approach with a hierarchical taxonomy of sound sources is proposed in [12]. In the design of our framework, we tried to have a flexible architecture that can support these models of top-down flow and hierarchical classification as well as traditional bottom-up processing.

2.1 Architecture

The system is implemented using a client-server architecture. The server written in C++, contains all the signal processing and pattern recognition modules optimized for performance. The client, written in Java, contains only the user interface code and communicates with the computation engine via sockets.

This breakdown has the advantage of decoupling the interface from the computation code and allows different interfaces to be built that use the same underlying computational functionality. For example, the server can be accessed by different graphical user interfaces, scripting tools, web crawlers, etc.

Special attention was given to abstracting the audio analysis process using object-oriented programming techniques. Abstract classes are used to provide a common API for the building blocks of the system. and inheritance is used to factor out common operations. The main classes of the system can roughly be divided in two categories: process-like objects and data-structure-like objects.

Process objects:

Transformations are the low-level signal processing units used by the system. They take as input a frame of sound samples and output a transformation of that frame. Some examples are: power spectral density, cepstrum, windowing, digital filtering.

Features process a frame of sound samples and output a vector which unlike transformations is reduced significantly in dimensionality. They typically use sound transformations for their calculations. Because the output is a vector, more than one "physical" features can be combined. For example, both spectral centroid and rolloff involve the calculation of power spectral density and it is possible to bundle them together into one feature for increased performance or to use them separately for rapid prototyping.

Memories are circular buffers that hold previously calculated features for a limited time. They are used to compute means and variances of features over large windows without recomputing the features. They can have different sizes depending on the application.

Iterators break up a sound stream into frames. For each frame they use features and memories to calculate a feature vector. The result is a time-series of feature vectors which is called the feature map. Typically, there is a different iterator for each classification scheme. For example, a silence/non-silence iterator uses energy as a feature and has no memories. A more complicated iterator like the music/speech iterator uses 9 features and 2 memories (of different size) for feature calculation.

Classifiers take as input a feature vector and return its estimated class. They are trained using labeled feature maps.

Segmentors take as input feature maps and output a signal with peaks corresponding to segmentation boundaries.

Data structure objects:

Vectors are the basic data components of the system. They are float arrays tagged with sizes. Operator overloading is used for vector operations to avoid writing many nested loops for signal processing code. The operator functions are inlined and optimized. The resulting code is easy to read and understand without compromising performance.

Sound Data objects contain samples of audio as vectors with additional header information such as sample rate, channels, etc.

Feature Maps are time-series of feature vectors. Feature maps can be class labeled for evaluation and training.

Time Regions are time intervals tagged with annotation information.

Time Lines are lists of time regions.

Time Trees are arbitrary trees of time regions. They represent a hierarchical decomposition of audio into successively smaller segments (see Fig. 2).

All the objects contain methods to read/write them to files and transport them using the socket interface. For example a calculated feature map can be stored and used to evaluate different classifiers without having to redo the feature calculation for each classifier.

Although the objects form a natural bottom-up hierarchy, top-down flow of information can be expressed in the framework. As a simple example, a silence feature can be used by an iterator for music/speech to avoid calculating features on silent frames. Similarly hierarchical classification can be expressed using multiple iterators with different features.

2.2 Features

In our system we have implemented a number of the features that have been proposed in the literature. These features form a pool from which different algorithms can pick specific features depending on the specific task. Many classification algorithms, for example, must be invariant to loudness so they would not use energy as a feature. On the other hand, a segmentation algorithm would probably include energy as one of the features indicating a change in texture.

Some of the currently supported features, with references to systems that describe and use them, are:

Spectral Centroid is the balancing point of the spectrum. It can be calculated using

$$C = \frac{\sum_i i A_i}{\sum_i A_i} \tag{1}$$

where A_i is the the amplitude of frequency bin i of the spectrum [18, 7, 23, 15].

Spectral Moments are statistical measures that characterize the shape of the spectrum [7].

Spectral Flux is the 2-norm of the difference between the magnitude of the Short Time Fourier Transform (STFT) spectrum evaluated at two successive sound frames. The STFT is normalized in energy [18, 15].

Pitch is a pitch estimate for the frame and can be calculated using various different techniques [14].

Harmonicity is measure of how strong the pitch perception for a sound is [23]. It can also be used for voiced/unvoiced detection.

Mel-Frequency Cepstral Coefficients (MFCC) are commonly used in speech recognition [10, 18].They are a perceptually motivated compact representation of the spectrum [9].

Linear prediction (LPC) reflection coefficients are used in speech research as an estimate of the speech vocal tract filter [11].

Other features supported include Zero Crossings, RMS, Spectral Rolloff and others. For all these features means,variances and higher-order statistics over larger time windows can be calculated using memories. New features can easily be added to the system by writing only the code for computing the feature value from a frame of sound samples.

2.3 Classifiers

Currently, two statistical pattern recognition classifiers are implemented as part of the system. For a more complete description of these classifiers and statistical pattern recognition in general refer to [4].

The Gaussian (MAP) classifier assumes each class can be represented as a multi-dimensional normal distribution in feature space. A labeled data set is used to train the classifier by calculating the parameters for each particular class. This classifier is typical of parametric statistical classifiers that assume a particular form for the class probability density functions.

Unlike parametric classifiers, the K-Nearest-Neighbor classifier directly uses the training set for classification without assuming any mathematical form for the underlying class probability density functions. Each sample is classified according to the class of its nearest neighbor in the training data set. In K-NN, the K nearest-neighbors are calculated and voting is used to determine the class.

Due to the flexibility of the architecture, new and more advanced classifiers can easily be added. Gaussian Mixture and Neural Network classifiers are currently under development. The ability to have different classifiers allows trade-off between classification speed and accuracy depending on the application.

2.4 Framework Evaluation

The data used for evaluating the system consists of about 2 hours of audio data. There are 45 minutes of speech, 45 minutes of music, and about 30 minutes of mixed audio. Radio, live recordings of speech, compact disks and movies representing a variety of speakers and music styles were used as data sources.

As a test for the architecture a music/speech discriminator similar to the one described in [18, 15] was implemented. The implementation was used to test and refine the design of the framework. The discriminator runs in real-time on an SGI O2 workstation. Fig. 1 shows the layout and information flow of the implementation.

The performance of our system for music/speech discrimination is comparable to the recognition accuracy of current systems [18, 15] (90.1% for the K-NN(5) classifier). A direct comparison is impossible due to the differences in data sets and classifiers. We use a cross-validation testing framework [18] to ensure that the evaluation is not dependent on the particular test and training sets we have used.

In addition, an instrument identification system for steady-state tones similar to [7] has been implemented. Similarity retrieval as described in [23] is also supported.

3 Segmentation

Segmentation is based on the idea that transitions in audio texture will result in sudden changes of values in most of the features. This idea is consistent with research in psychoacoustics [3] where multiple perceptual cues are combined to indicate the onset of a new sound event.

The algorithm works in four stages:

1. A time series of feature vectors V_t is calculated by iterating over the sound file. Each feature vector can be thought of as a short description of the corresponding time-frame.

2. A distance-metric $\Delta_t = ||V_t - V_{t-1}||$ is calculated between successive frames of sound. In our implementation we use a Mahalonobis distance. It is defined by

$$D(x,y) = (x-y)^T \Sigma^{-1} (x-y) \tag{2}$$

   where Σ is the feature covariance matrix calculated from the training set. Other distance metrics, possibly using relative feature weighting, can also be used.

3. The derivative $\frac{d\Delta_t}{dt}$ of the distance signal is taken. Thresholding is then used for finding the peaks of the result. The derivative of the distance will be low for slowly changing textures and high during sudden transitions. Therefore the peaks roughly correspond to texture changes.

4. Peaks are picked using simple heuristics and used to create the segmentation of the signal into time regions. As a heuristic example, a minimum duration between successive peaks can be set to avoid small regions. The result is stored as a time line (i.e a list of time intervals) and can be used for browsing and annotating (see Fig. 2).

3.1 Combining segmentation and classification

Most of the classification methods proposed in the literature report improved performance if the classification results are integrated over larger time windows. However, using fixed size integration windows blurs the transition edges between classes. Usually, test data consists of files that do not contain transitions to simplify the evaluation; therefore this problem does not show up. In real world data, however, transitions exist and it is important to preserve them.

The described segmentation method provides a natural way of breaking up the data into regions based on texture. These regions can then be used to integrate classification results. That way, sharp transitions are preserved and the classification performance is improved because of the integration. Initial experiments in a number of different sound files confirm this fact. A more detailed quantitative evaluation of how this method comapres to fixed-window integration is under way.

3.2 Segmentation Results

The method has been tested using various sound files from our data set. Representative examples include detecting a guitar solo entrance or a cymbal crash, change of speaker, transitions from music to speech, and musical structure for cyclic (ABAB type) pop songs.

The segmentation results are difficult to evaluate quantitatively because they depend on the choice of features and parameter values like the memory-size, peak-threshold, distance-metric and others. Typically, the resulting regions are perceptually meaningful. For now, fine-tuning of the parameters is done by the user (or the programmer) depending on the desired result. For example lowering the peak-threshold results into more and smaller regions. Automatic parameter adjustment is investigated.

4 User Interface

The interface looks like a typical tape-recorder style waveform-editor. However, in addition to the typical play, fast-forward, rewind and stop buttons it allows skipping by either user defined fixed duration blocks or time lines containing regions of variable duration. These time lines are created either by hand or automatically using the segmentation method described above.

Skipping and annotating using regions is much faster than manual annotation, in the same way that finding a song on a CD is much faster than finding it on a tape.

The user can select a region and retrieve similar sounds. Another possibility is to classify the region using one of the available classification schemas like the music/speech discriminator. Finally, each time region can be annotated by multiple keywords.

In addition, the user can combine time regions to form a time tree that can be used for multi-resolution browsing and annotation. The tree captures the hierarchical nature of music pieces, and therefore can be used for musical analysis.

5 Future work

On the interface side, we plan to support multiple time lines and fast keyword search in annotations. Another interesting application of our segmentation scheme is audio thumbnails. For each region, a characteristic segment has to be selected. These segments are then used to create a shorter summary version of the original sound file.

On the computational side, we are investigating using a more perceptually accurate front end to the system. A number of computational models of the ear physiology have been proposed in the literature [21, 20] and can be used as basis for feature calculation.

Most attempts to build music analysis systems in the past have tried to extract content by first transcribing the music into symbolic notation and then using music theory to characterize it. This approach has been challenged by [13, 16]. Current systems try to analyze structure and content directly from features calculated from the audio signal. Such systems can easily be implemented and evaluated using our framework. As a simple example of music analysis, the structure of cyclic pop songs can be revealed using our segmentation scheme. We believe that a combination of our segmentation scheme with beat tracking methods [17] can offer significant information for music style identification and music analysis.

Finally, we plan to write a web crawler that will automatically create segmentation time lines and annotations for files on the Web.

6 Summary

We designed and implemented a flexible framework for building and integrating audio analysis tools. A number of existing techniques were implemented and tested to evaluate the framework. A new method for audio segmentation based on texture is presented. This method combined with the analysis tools and using a semi-automatic user interface offers significant improvements for audio searching, annotation and browsing. In addition, it can be used to improve classification performance by using the results of the segmentation to adaptively change the classification integration window size.

References

[1] B Arons. Speechskimmer: A system for interactively skimming recorded speech. *ACM Transactions Computer Human Interaction*, 4:3–38, 1997. http://www.media.mit.edu/people/barons/papers/ToCHI97.ps.

[2] J Boreczky and L Wilcox. A hidden markov model framework for video segmentation using audio and image features. *Proc. Int.Conf on Acoustics,Speech and Signal Processing Vol.6*, pages 3741–3744, 1998.

[3] A Bregman. *Auditory Scene Analysis*. MIT Press, 1990.

[4] R Duda and P Hart. *Pattern Classification and Scene Analysis*. John Wiley & Sons, 1973.

[5] D Ellis. *Prediction-driven computational auditory scene analysis*. PhD thesis, MIT Dept. of Electrical Engineering and Computer Science, 1996.

[6] J Foote. An overview of audio information retrieval. *ACM Multimedia Systems*, 7:2–10, 1999.

[7] I Fujinaga. Machine recognition of timbre using steady-state tone of acoustic instruments. *Proc. ICMC 98*, pages 207–210, 1998.

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

[9] M Hunt, M Lennig, and P Mermelstein. Experiments in syllable-based recognition of continuous speech. In *Proc. 1996 ICASSP*, pages 880–883, 1980.

[10] D Kimber and L Wilcox. Acoustic segmentation for audio browsers. *Proc. Interface Conference (Sydney, Australia 96)*, 1996.

[11] J Makhoul. Linear prediction: A tutorial overview. *Proc.IEEE*, 63:561–580, April 1975.

[12] K Martin. Toward automatic sound source recognition: identifying musical instruments. In *NATO Computational Hearing Advanced Study Institute*. Il Ciocco IT, 1998.

[13] K Martin, E Scheirer, and B Vercoe. Musical content analysis through models of audition. In *Proc.ACM Multimedia Workshop on Content-Based Processing of Music*, Bristol, UK, 1998.

[14] L Rabiner, M Cheng, A Rosenberg, and C McGonegal. A comparative performance study of several pitch detection algorithms. *IEEE Trans. Acoust., Speech, and Signal Process.*, ASSP-24:399–417, October 1976.

[15] S Rossignol, X Rodet, et al. Features extraction and temporal segmentation of acoustic signals. *Proc. ICMC 98*, pages 199–202, 1998.

[16] E Scheirer. Bregman's chimerae: Music perception as auditory scene analysis. In *Proc.International Conference on Music Perception and Cognition*, Montreal, 1996.

[17] E Scheirer. Tempo and beat analysis of acoustic musical signals. *J.Acoust.Soc.Am*, 103(1):588,601, Jan 1998.

[18] E Scheirer and M Slaney. Construction and evaluation of a robust multifeature speech/music discriminator. *IEEE Transactions on Acoustics, Speech and Signal Processing (ICASSP'97)*, pages 1331–1334, 1997.

[19] M Slaney. A critique of pure audition. *Computational Auditory Scene Analysis*, 1997.

[20] M Slaney and R Lyon. A perceptual pitch detector. In *Proceedings of the 1990 International Conference on Acoustics, Speech and Signal Processing (ICASP)*, pages 357–360, Albuquerque, NM, 1990. IEEE.

[21] M Slaney and R Lyon. On the importance of time-a temporal representation of sound. In M Cooke, B Beet, and M Crawford, editors, *Visual Representations of Speech Signals*, pages 95–116. John Wiley & Sons Ltd, 1993.

[22] C van Rijsbergen. *Information retrieval.* Butterworths, London, 2nd edition, 1979.

[23] E Wold, T Blum, D Keislar, and J Wheaton. Content-based classification, search and retrieval of audio. *IEEE Multimedia*, 3(2):27–36, 1996.

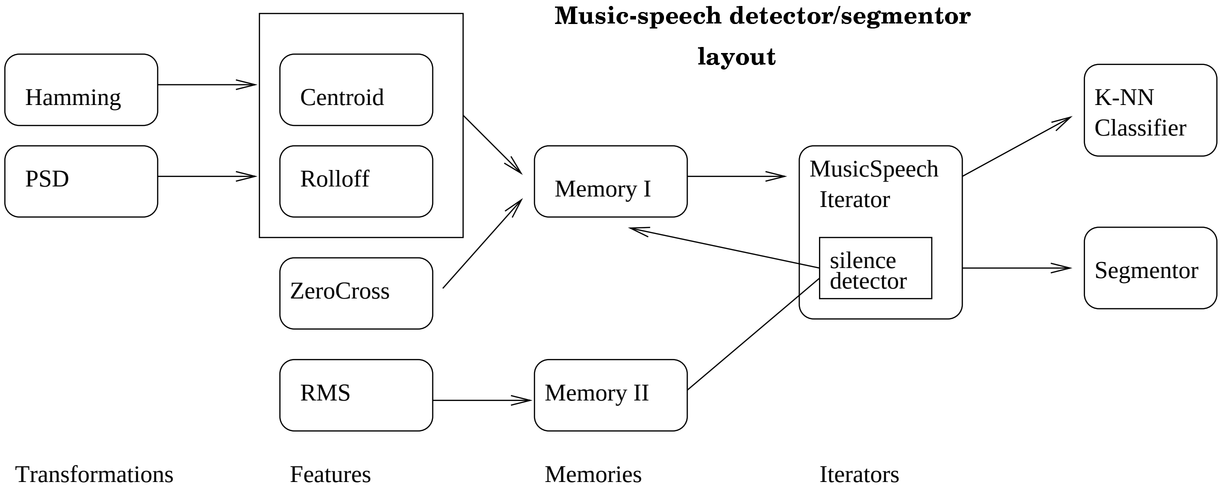

Figure 1: Music Speech Detector

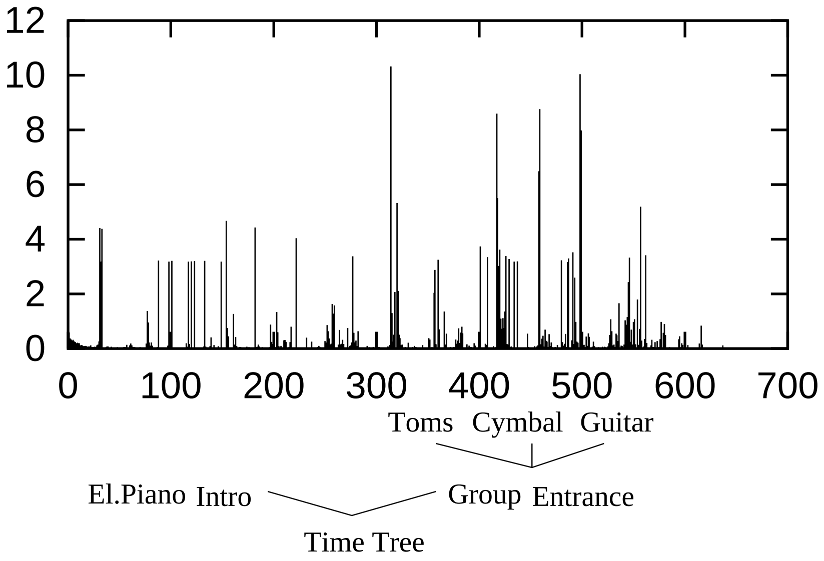

Figure 2: Segmentation peaks and corresponding time tree