MANIPULATION, ANALYSIS AND RETRIEVAL SYSTEMS FOR AUDIO SIGNALS

GEORGE TZANETAKIS

A DISSERTATION
PRESENTED TO THE FACULTY
OF PRINCETON UNIVERSITY
IN CANDIDACY FOR THE DEGREE
OF DOCTOR OF PHILOSOPHY

RECOMMENDED FOR ACCEPTANCE
BY THE DEPARTMENT OF
COMPUTER SCIENCE

JUNE 2002

To my grandfather Giorgos Tzanetakis whose passion for knowledge and love for all living things will always be an inspiration for me.

To my parents Panos and Anna and my sister Flora for their support and love.

Abstract

Digital audio and especially music collections are becoming a major part of the average computer user experience. Large digital audio collections of sound effects are also used by the movie and animation industry. Research areas that utilize large audio collections include: Auditory Display, Bioacoustics, Computer Music, Forensics, and Music Cognition.

In order to develop more sophisticated tools for interacting with large digital audio collections, research in Computer Audition algorithms and user interfaces is required. In this work a series of systems for manipulating, retrieving from, and analysing large collections of audio signals will be described. The foundation of these systems is the design of new and the application of existing algorithms for automatic audio content analysis. The results of the analysis are used to build novel 2D and 3D graphical user interfaces for browsing and interacting with audio signals and collections. The proposed systems are based on techniques from the fields of Signal Processing, Pattern Recognition, Information Retrieval, Visualization and Human Computer Interaction. All the proposed algorithms and interfaces are integrated under MARSYAS, a free software framework designed for rapid prototyping of computer audition research. In most cases the proposed algorithms have been evaluated and informed by conducting user studies.

New contributions of this work to the area of Computer Audition include: a general multifeature audio texture segmentation methodology, feature extraction from mp3 compressed data, automatic beat detection and analysis based on the Discrete Wavelet Transform and musical genre classification combining timbral, rhythmic and harmonic features. Novel graphical user interfaces developed in this work are various tools for browsing and visualizing large audio collections such as the Timbregram, TimbreSpace, GenreGram, and Enhanced Sound Editor.

Acknowledgments

First of all I would like to thank Perry Cook for being such a great advisor. Perry has always been supportive of my work and a constant source of new and interesting ideas. Large parts of the research described in this thesis were written after Perry said “I have a question for you. How hard would it be to do ...”. In addition I would like to thank the members of my thesis committe: Andrea LaPaugh, Ken Steiglitz, Tom Funkhauser and Paul Lansky who provided valuable feedback and support especially during the last difficult months of looking for a job and finishing my PhD.

Kimon has motivated me to walk and run in the woods, clearing my brain as I follow his wagging tail. Emily provided support, soups and stews. Lujo Bauer, Georg Essl, Neophytos Michael, and Kedar Swadi and all the other graduate students at the Computer Science department have been great friends and helped me many times in many different ways.

During my years in graduate school it was my pleasure to collaborate with many undergraduate and graduate students doing projects related to my research and in many cases using MARSYAS the computer audition software framework I have developed. These include John Forsyth, Fei Fei Li, Corrie Elder, Doug Turnbull, Anoop Gupta, David Greco, George Tourtellot, Andreye Ermolinskyi, and Ari Lazier.

The two summer internships I did at the Computer Human Interaction Center (CHIC) at SRI International and at Moodlogic were great learning experiences and provided with valuable knowledge. I would like to especially thank Luc Julia and Rehan Khan for being great people to work with.

Finally, research costs and I was fortunate enough to have support from the following funding sources: NSF grant 9984087, New Jersey Commission on Science and Technology grant 01-2042-007-22, Intel and Arial Foundation.

Contents

Abstract iv

1 Introduction **1**

1.1 Motivation and Applications 1

1.2 The main problem 3

1.3 Current state 3

1.4 Relevant Research Areas 4

1.5 Guiding principles and observations 7

1.6 Overview 11

1.7 Related work 14

1.8 Main Contributions 18

1.9 A short tour 20

2 Representation **24**

2.1 Introduction 24

2.2 Related Work 26

2.2.1 Contributions 29

2.3 Spectral Shape Features 31

2.3.1 STFT-based features 32

2.3.2 Mel-Frequency Cepstral Coefficients 33

2.3.3 MPEG-based features . . . 34
2.4 Representing sound "texture" . . . 38
2.4.1 Analysis and texture windows . . . 39
2.4.2 Low Energy Feature . . . 40
2.5 Wavelet transform features . . . 40
2.5.1 The Discrete Wavelet Transform . . . 40
2.5.2 Wavelet features . . . 41
2.6 Other features . . . 42
2.7 Rhythmic Content Features . . . 43
2.7.1 Beat Histograms . . . 43
2.7.2 Features . . . 46
2.8 Pitch Content Features . . . 49
2.8.1 Pitch Histograms . . . 49
2.8.2 Features . . . 51
2.9 Musical Content Features . . . 52
2.10 Summary . . . 52

3 Analysis 54
3.1 Introduction . . . 54
3.2 Related Work . . . 55
3.2.1 Contributions . . . 58
3.3 Query-by-example content-based similarity retrieval . . . 59
3.4 Classification . . . 60
3.4.1 Classifiers . . . 61
3.4.2 Classification schemes . . . 64
3.4.3 Voices . . . 64
3.5 Segmentation . . . 67

3.5.1 Motivation 67
3.5.2 Methodology 69
3.5.3 Segmentation Features 71
3.6 Audio thumbnailing 71
3.7 Integration 72
3.8 Summary 72

4 Interaction **74**
4.1 Introduction 74
4.2 Related Work 76
4.2.1 Beyond the Query-by-Example paradigm 79
4.2.2 Contributions 81
4.3 Terminology 81
4.4 Browsers 83
4.4.1 Timbrespace Browser 83
4.4.2 Timbregram viewer and browser 86
4.4.3 Other Browsers 91
4.5 Enhanced Audio Editor 92
4.6 Monitors 94
4.6.1 GenreGram 94
4.6.2 TimbreBall monitor 95
4.6.3 Other monitors 95
4.7 Audio-based Query Interfaces 96
4.7.1 Sound Sliders 96
4.7.2 Sound palettes 98
4.7.3 Loops 98
4.7.4 Music Mosaics 98

4.7.5 3D sound effects generators 99
4.8 MIDI-based Query Interfaces 100
4.8.1 The groove box 101
4.8.2 The tonal knob 102
4.8.3 Style machines 102
4.9 Integration 103
4.10 The Princeton Display Wall 104
4.11 Summary 106

5 Evaluation 107

5.1 Introduction 107
5.2 Related Work 108
5.2.1 Contributions 108
5.3 Collections 109
5.4 Classification evaluation 110
5.4.1 Cross-Validation 111
5.5 Segmentation experiments 111
5.6 MPEG-based features 117
5.6.1 Music/Speech classification 118
5.6.2 Segmentation evaluation 119
5.7 DWT-based features 120
5.8 Similarity retrieval evaluation 121
5.9 Automatic Musical Genre Classification 123
5.9.1 Results 123
5.9.2 Human performance for genre classification 129
5.10 Other experimental results 130
5.10.1 Audio Thumbnailing 130

5.10.2 Annotation 130
5.10.3 Beat studies 131
5.11 Summary 133

6 Implementation 134

6.1 Introduction 134
6.2 Related Work 136
6.3 Architecture 138
6.3.1 Computation Server 138
6.3.2 User Interface Client - Interaction 142
6.4 Usage 143
6.5 Characteristics 144

7 Conclusions 146

7.1 Discussion 146
7.2 Current Work 147
7.3 Future work 149
7.3.1 Other directions 154
7.4 Final Thoughts 156

A 157

A.1 Introduction 157
A.2 Short Time Fourier Transform 157
A.3 Discrete Wavelet Transform 158
A.4 Mel-Frequency Cepstral Coefficients 159
A.5 Multiple Pitch Detection 160
A.6 Gaussian classifier 162
A.7 GMM classifier and the EM algorithm 162

A.8 K-Means 164
A.9 KNN classifier 165
A.10 Principal Component Analysis 165

B **167**
B.1 Glossary 167

C **170**
C.1 Publications 170

D **173**
D.1 Web Resources 173

List of Figures

1.1 Relevant Research Areas 5
1.2 Computer Audition Pipeline 13

2.1 Magnitude Spectrum 31
2.2 Sound textures of news broadcast 38
2.3 Beat Histogram Calculation Flow Diagram 44
2.4 Beat Histogram Example 47
2.5 Beat Histogram Examples 48

3.1 Musical Genres Hiearachy 65
3.2 Classification Scemes Accuracy 66
3.3 Feature-based segmentation and classification 73

4.1 2D TimbreSpace of Sound Effects 84
4.2 3D TimbreSpace3D of Sound Effects 85
4.3 3D TimbreSpace of orchestral music 86
4.4 Timbregram superimposed over waveform 88
4.5 Timbregrams of music and speech 89
4.6 Timbregrams of classical, rock and speech 90
4.7 Timbregrams of orchestral music 90
4.8 Enhanced Audio Editor I 92

4.9 Enhanced Audio Editor II 93
4.10 Timbreball and GenreGram monitors 94
4.11 Sound sliders and palettes. 97
4.12 3D sound effects generators. 99
4.13 Groove box 101
4.14 Indian music style machine 102
4.15 Integration of interfaces 103
4.16 Marsyas on the Display Wall 105

5.1 Musical Genres Hiearachy 109
5.2 Histograms of subject agreement and learning curve for mean segment completion time 116
5.3 1,2,3 and 4,5,6 are the means and variances of MPEG Centroid,Rolloff and Flux. Low Energy is 7. 118
5.4 DWT-based feature evaluation based on audio classification 120
5.5 Web-based Retrieval User Evaluation 121
5.6 Results of User Evaluation 122
5.7 Classification accuracy percentages (RND=random,RT=real time, WF=whole file) 125
5.8 Effect of texture window size to classification accuracy 128
5.9 Synthetic Beat Patterns (CY = cymbal, HH = hi-hat, SD = snare drum, BD = base drum, WB = wood block, HT/MT/LT = tom-toms) 132

6.1 Music Speech Discriminator Architectural Layout 140
6.2 Segmentation peaks and corresponding time tree 142
6.3 Marsyas User Interface Component Architecture 143

A.1 Discrete Wavelet Transform Pyramidal Algorithm Schematic 159

A.2 Summing of FFT bins for MFCC filterbank calculation 160
A.3 Multiple Pitch Analysis Block Diagram 161
A.4 Schematic of Principal Component extraction 166

Chapter 1

Introduction

The process of preparing programs for a digital computer is especially attractive, not only because it can be economically and scientifically rewarding, but also because it can be an aesthetic experience much like composing poetry or music.

Donald E. Knuth, The Art of Computer Programming

(especially when you write programs to analyze audio signals)

1.1 Motivation and Applications

th Developments in hardware, compression technology and network connectivity have made digital audio and especially music a major part of the everyday computer user experience and a significant portion of web traffic. The media and legal attention that companies like Napster have recently received, testifies to this increasing importance of digital audio. It is likely that in the near future the majority of recorded music in human history will be available digitally. As an indication of the magnitude of such a collection currently the number of cd tracks in retail is approximately 3 million. It is still unclear what the business models for this new brave new music world will be. However, one thing that is certain is the need to interact effectively with increasingly large digital audio collections.

In addition to digital music distribution, the need to interact with large audio collections arises in other applications areas as well. Most of the movies produced today use a large number of sounds taken from large libraries of sound effects (for example the BBC Sound Effects Library contains 60 compact disks). Whenever we hear a door closing, a gun shot or a telephone ringing in a movie it almost always originates from one of those libraries. Even larger numbers of sound effects are used in the animation industry. The importance and usage of audio in computer games is also increasing. Moreover, composers, DJs, and arrangers frequently utilize collections of sounds to create musical works. These sounds typically originate from synthesizers, samplers and musical instruments recorded live.

In addition to applications related to the entertainment industry, the need to interact with large audio collections arises in many scientific disciplines. Auditory Display [1] is a new research area that focuses on the use of non-speech audio to convey information. Specific areas include the sonification of scientific data, spatial sound, and the use of auditory icons in user interfaces. 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. Audio analysis tools have also applications in Forensics which is the use of science and technology to investigate and establish facts in criminal or civil courts of law. For example the make of a particular car or the identity of a speaking person can be automatically identified from a recording. Other fields where working with audio collections is important are Bioacoustics, Psychoacoustics and Music Cognition. Bioacoustics is the study of sounds produced by animals. Psychoacoustics studies the perception of different types of sound by animals of the human species and Music Cognition looks specifically at how we perceive and understand music.

This indicative but by no means exhaustive list of application areas from industry and academia shows the need for tools to manipulate, analyze and retrieve audio signals from large digital audio collections.

[1] `http://www.icad.org`

1.2 The main problem

To summarize, the main problem that this work tries to address is the need to manipulate, analyze and retrieve from large digital audio collections, with special emphasis on musical signals. In order to achieve this goal a series of existing and novel automatic computer audition algorithms were designed, developed, evaluated, and applied directly to create novel content and context aware audio graphical user interfaces for interacting with large audio collections.

1.3 Current state

One of the main limitations of current multimedia software systems is that they have very little understanding of the data they process and store. For example an audio editing program represents audio as an unstructured monolithic block of digital samples. In order to develop new effective tools for interacting with audio collections it is important to have information about the content and context of audio signals. Content refers to any information about an audio signal and its structure. Some examples of content information are: knowing a specific section of a jazz music piece corresponds to the saxophone solo, identifying a bird from its song, finding similar sounds to a particular walking sound in a collection of sound effects for movies. Context refers to information about an audio signal that is dependent upon a larger collection to which it belongs. As an example of context information in a large collection of all musical styles , two female singer Jazz pieces would be similar whereas in a collection of vocal Jazz they might be quite dissimilar. Therefore their similarity is context information as it depends on the particular audio collection. Another example of context information is the notion of a “fast” musical piece which is different for different musical genres therefore depends on the particular context.

Obviously in order to achieve this sensitivity to the content and context of audio collections more high level information than what is currently used is required. The most common approach to address this problem has been to manually annotate audio signals with additional information (for example filenames and ID3 tags of mp3 files). However this approach has several limitations: it requires significant user time (there are commercial companies that pay full-time experts to perform this annotation), does not scale well, and can only provide specific types of information that are typically categorical in nature such as music genre, mood, or energy level. In addition, the maintenance of all this information requires significant infrastructure and quality assurance in order to ensure that the annotation results are consistent and accurate. Therefore algorithms and techniques that are capable of extracting such information automatically are desired. In this thesis, a variety of such algorithms and techniques will be proposed and will be used to create novel tools for structuring and interacting with large audio collections.

1.4 Relevant Research Areas

This work falls under the general research area of *Computer Audition* (CA). The term *Computer Audition* is used in a similar fashion to the term *Computer Vision* and it includes any process of extracting information from audio signals. Other terms used in the literature are *Machine Listening* and *Auditory Scene Analysis*. [2]

Although *Speech Recognition* can be considered part of CA, for the purposes of this thesis, CA will refer to techniques that although possibly process speech signals, they do not perform any language modeling. For example, speaker/singer identification and music/speech discrimination will be included as they can be performed by a human who

[2] I prefer the term *Computer Audition* because by being more general is applicable to a broader set of problems. *Machine Listening* implies a high level perceptual activity whereas the term *Auditory Scene Analysis* refers to the recognition of different sound producing objects in a scene. Although there is no common agreement about the terminology I consider these two other terms subtopics of *Computer Audition.*

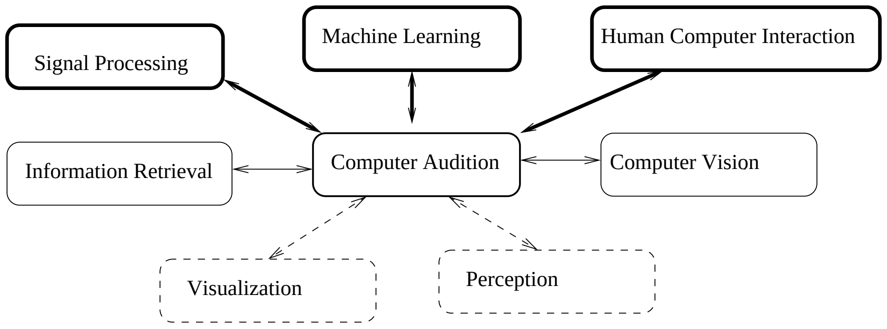


Figure 1.1: Relevant Research Areas

might not know the language spoken but *Speech Recognition* and *Speech Synthesis* will not be covered as they rely on language models. Although research in aspects of *Computer Audition* has existed for a long time only recently (after 1996) there has been significant research output. [3]

Some of the reasons for this late interest are:

- Developments in hardware that made feasible computations over large collections of audio signals
- Audio compression such as the MPEG audio compression standard (.mp3 files) which made digital music distribution over the internet a reality
- A shift from symbolic methods and small "toy" examples to statistical methods and large collections of "real world" audio signals (also a trend in the general *Artificial Intelligence* community)

Despite the fact that *Computer Audition* is a relatively recent research area still in its infancy there are many existing well-established research areas where interesting ideas and tools can be found and adapted for the purpose of extracting information from audio signals.

Figure 1.1 shows the main research areas that are relevant to this work. The thickness of the lines is indicative of the importance of each area to this particular work rather

[3] My graduate studies began in 1997 and therefore I have been very lucky as I was able to track most of the new research in Computer Audition and actively participate in its evolution.

than to the area of *Computer Audition* in general. Audio and especially music signals are complex data-intensive, time-varying signals with unique characteristics that pose new challenges to these relevant research areas. Therefore, research in CA provides a good way to test, evaluate and create new algorithms and tools in all those research areas. This reciprocal relation is indicated in figure 1.1 with the use of connecting arrows pointing in both directions. In the following paragraphs short descriptions of these research ares and their connection to Computer Audition are provided. In addition a representative reference textbook (by no means the only one or the best one) for each specific area is provided.

Digital Signal Processing [97] is the computer manipulation of analog signals (commonly sounds or images) which have been converted to digital form (sampled). The various techniques of Time-Frequency analysis developed for processing audio signals in many cases originally developed for Speech Processing and Recognition [98], are of special importance for this work.

Machine Learning techniques allow a machine to improve its performance based on previous results and training by humans. More specifically pattern recognition machine learning techniques [30] aim to classify data (patterns) based on either a priori knowledge or on statistical information extracted from the patterns. The patterns to be classified are usually groups of measurements or observations, defining points in an appropriate multidimensional space.

Computer Vision [9] is a term used to describe any algorithm and system that extracts information from visual signals such as image and video. Of specific interest to this work is research in video analysis as both video and audio are temporal signals.

Human Computer Interaction (HCI) research is becoming an increasingly important area of Computer Science research [114]. Graphical user interfaces (GUIs) enable the creation of powerful tools that combine the best of human and machine capabilities. Of specific interest to this work is the field of *Information Visualization* [120, 34] that deals with the visual presentation of various types of information in order to better understand its structure and characteristics.

Information Retrieval [137] familiar from the popular web search engines such as AltaVista (`www.altavista.com`) and Google (`www.google.com`), refers to techniques for searching and retrieving text documents based on user queries. More lately techniques for Multimedia Information Retrieval such as content-based image retrieval systems [37] have started appearing.

Last but not least is *Perception* research. The human perceptual system has unique characteristics and abilities that can be taken into account to develop more effective computer systems for processing sensory data. For example both in images and sound, knowledge of the properties of the human perceptual systems has led to powerful compression methods such as JPEG (for images) and MPEG (for video and audio). A good introduction to psychoacoustics (the perception of sounds) using computers is [21].

1.5 Guiding principles and observations

Given the current state of tools for interacting with large audio collections several directions for improvement are evident. Based on these points, the following guiding principles characterize this work and differentiate it from the majority of existing research:

- **Emphasis on fundamental problems and "real world" signals**

  Many existing systems are too ambitious in the sense that they solve some hard problem such as polyphonic transcription but only for very limited artificially constructed signals ("toy problems"). In contrast in this work the emphasis is on solving fundamental problems that are applicable to arbitrary "real world" audio signals such music pieces stored as mp3 files.

- **Develop new and improve existing computer audition algorithms**

In order to design and develop new algorithms in any field it is important to first understand, implement, compare and improve existing algorithms. Therefore we have tried to include and improve many existing CA algorithms in addition to the newly designed algorithms.

- **Integration of different analysis techniques**

  A large percentage of existing work in CA deals with only a specific analysis problem. However in many cases the results of different analysis techniques can be integrated improving the resulting analysis. For example segmentation can be used to reduce classification errors. A consequence of this desire for integration has been the creation of a general and flexible software infrastructure for CA research rather than solving each small individual problem separately.

- **Content and Context aware user interfaces based on automatic analysis**

  Existing CA work emphasizes automatic algorithms and is rarely concerned with how users can interact with these algorithms. Human computer interaction is a very important aspect in the design of an effective system. In this thesis our goal is to provide combine the abilities of humans and computers by creating novel user interfaces to interact with audio signals and collections. Unlike existing tools these interfaces are aware of content and context information about the audio signals they present and this information is extracted using automatic analysis algorithms.

- **Interacting with large collections rather than individual files**

  In both the developed algorithms and interfaces the emphasis is on working with large collections rather than individual files. So, for example, large part of the work in interface design is concerned with browsing large collections while the automatic analysis is heavily based on statistical learning techniques that utilize large data sets for training.

- **Automatic and user-based evaluation**

  Evaluation of systems that try to imitate human sensory perception is difficult because there is no single well-defined correct answer. However it is feasible and it is important for the design and development of effective CA algorithms and systems. Because CA is a relatively recent and still developing field many existing systems are proposed without detailed evaluation (for example only a few examples where the algorithms works are presented). In this thesis, an effort has been made, to evaluate the proposed techniques statistically using large data sets and with user experiments when that is required.

- **Real-time performance** Computer audition computations are numerically-intensive and stress current hardware to its limits. In order to have desired real-time performance and interactivity in the user interfaces careful choices have to be made both at the algorithmic and the implementation level. Unlike certain proposed algorithms that take hours to analyze a minute of audio, our emphasis has been on fast algorithms and efficient implementation that enables real-time feature extraction, analysis and interaction.

Although some of these design guidelines have been used in several proposed systems to the best of our knowledge no existing system has combined all of them. In addition the following simple, informal but important observations are some way or another behind many of the ideas described in this thesis.

- **Automatic techniques with errors can be made useful.**

  Automatic audio analysis techniques are still far from perfect and in many cases have large percentages of errors. However their results can be significantly improved using various techniques such as: statistics, integration of different analysis techniques, user interfaces and utilizing the errors as as source of information. For example although short time automatic beat detection is not accurate, a good representation of the overall beat of a song can be collected using statistics (see Beat Histograms, Chapter 2). In addition the failure of the beat detection is a source of information for musical genre classification (for example beat detection works better for rock and rap music than for classical or ambient). Although this observation is relatively simple, it leads to many interesting ideas that will be explored in the following chapters.

- **Avoid solving a simple problem by trying to solve a harder one**

  Although this observation sounds trivial it is not taken into account in a lot of existing work. For example a lot of work in audio segmentation assumes an initial classification. However classification is a harder problem as it involves learning both in humans and machines (there is no easy way to prove such an assertion, and therefore some people might disagree; however it will be shown that automatic segmentation without classification is feasible). Another example is the work in polyphonic transcription, which is still an unsolved problem, for the purpose of retrieval and classification which can be solved using statistical methods.

- **Use the software you write**

  In order to evaluate the algorithms and tools you develop it is important to use them and the most easily accessible user is yourself. An effort has been made in this work to develop working prototypes that are subsequently used to evaluate and improve the developed algorithms, creating that way a self-improving feedback loop.

The way these guiding principles and observations influence the ideas described in this thesis will become clearer in the following chapters that described in detail the developed algorithms and systems.

1.6 Overview

As in every PhD thesis my primary goal is to present in a clear and accurate manner my research work. In addition I would also like my thesis to serve as a good introduction to the field of Computer Audition in general and its current state of art. Because most of the research is relatively recent and I have implemented many of the existing algorithms proposed in the literature I hope that my goal is achievable. Ideally I would like this thesis to be a good starting point for someone entering the field of Computer Audition. Another side-effect of this secondary goal is the extended bibliography. Unlike more mature fields were there are good survey papers that can serve as starting points, Computer Audition is a relatively new field and it is possible (and I have attempted to do so) to cite the majority of directly relevant publications.

Toward this goal the structure of this thesis follows a form similar to a hypothetical Computer Audition textbook. For presentation purposes I like to divide Computer Audition in the following stages:

- **Representation - Hearing**

  Audio signals are data-intensive time-domain signals and are stored (in their basic uncompressed form) as series of numbers corresponding to the amplitude of the signal over time. Although this representation is adequate for transmission and reproduction of arbitrary waveforms it is not appropriate for analysing and understanding audio signals. The way we perceive and understand audio signals as humans is based on and constrained by our auditory system. It is well known that the early stages of the human auditory system (HAS) decompose incoming sound wave into different frequency bands. In a similar fashion, in Computer Audition, Time-Frequency analysis techniques are frequently used for representing audio signals. The representation stage of the Computer Audition pipeline refers to any algorithms and systems that take as input simple time-domain audio signals and create a more compact, information-bearing representation. The most relevant research area to this stage is Signal Processing.

- **Analysis - Understanding**

  Once a good representation is extracted from the audio signal various types of automatic analysis can be performed. These include similarity retrieval classification, clustering, segmentation, and thumbnailing. These higher-level types of analysis typically involve memory and learning both for humans and machines. Therefore Machine Learning algorithms and techniques are important for this stage.

- **Interaction - Acting**

  When the signal is represented and analyzed by the system, the user must be presented with the necessary information and act according to it. The algorithms and systems of this stage are influence by ideas from Human Computer Interaction and deal with how information about audio signals and collections can be presented to the user and what types of controls for handling this information are provided.

This division into stages will be used to structure this thesis. It is important to emphasize that the boundaries between these stages are not clear cut. For example a very complete representation (such as a musical score) makes analysis easier than a less detailed representation (such as feature vectors). However feature vectors can be easily calculated while the automatic extraction of musical scores (polyphonic transcription) is still an unsolved problem. Although these stages are presented separately for clarity, in any complete system they are all integrated. Additionally the design of algorithms and systems in each stage has to be informed and constrained by the design choices for the other stages. This becomes evident in evaluation and implementation that typically can not be applied to each stage separately but have to take all of them into account. Figure 1.2 shows the stages of the Computer Audition Pipeline and representative examples of algorithms and systems for

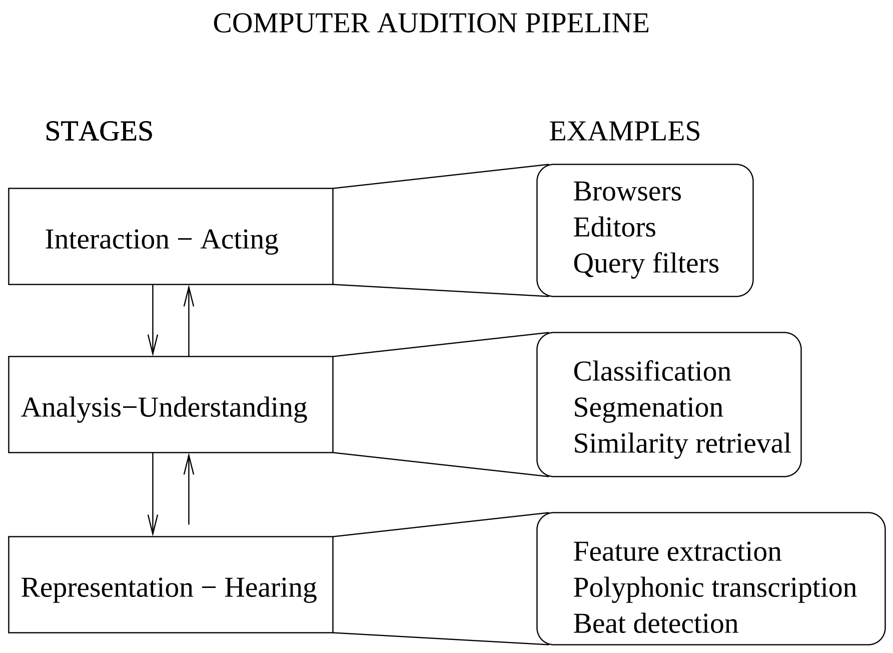


Figure 1.2: Computer Audition Pipeline

each stage. Although in most existing systems information typically flows bottom-up both directions of flow are important as has been argued in [115, 31].

This thesis is structured as follows: Chapter 1 (which you are currently reading) provides a high-level overview of the thesis, the motivation behind it and how it fits in the context of related work. Chapter 2 is about representation and describes various feature extraction techniques that create representations for different aspects of musical and audio content. The analysis stage is covered in Chapter 3 where techniques such as musical genre classification, sound "texture" segmentation and similarity retrieval are described. Chapter 4 corresponds to the interaction stage and is about various novel content and context aware user interfaces for interacting with audio signals and collections. The evaluation of the proposed algorithms and systems is described in Chapter 5. The reason evaluation is presented in a separate chapter is because in most cases it involves many different algorithms and systems from all the CA stages. For example in order to evaluate MPEG-based features, both classification and segmentation results are obtained through the use of developed graphical user interfaces. Chapter 6 talks about the implementation and architecture of our system. The thesis ends with Chapter 7 which summarizes the main thesis contributions and gives directions for future research. Each chapter contains a section about related work and the specific contributions of this work in each area. Standard algorithms techniques that are important for this work are described in more detail in Appendix A. A glossary of the terminology used in this thesis is provided for quick reference at Appendix B. Appendix C presents the publications that constitute the work presented in this thesis in chronological order and Appendix D provides some web resources related to this research.

For the remainder of the thesis, it is useful to have in mind the following hypothetical scenario that will very likely become reality in the near future. Suppose all of recorded music in the world is stored digitally in your computer and you want to be able to manipulate, analyze and retrieve from this large collection of audio signals. Although, in many cases, the techniques described in this work are also applicable to other types of audio signals such as sound effects and isolated musical instrument tones, the main emphasis will be on musical signals. So if there is no other indication in the text, the reader can assume that signals of interest will be musical signals. *Marsyas* is the name of the free software framework for Computer Audition research developed as part of this thesis under which all the described algorithms and interfaces are integrated.

1.7 Related work

In this section, pointers to important related work will be provided. The emphasis is on representative and influential publications as well as descriptions of complex Complex Audition systems rather than than individual techniques or algorithms. More detailed and complete references are provided in the related work section of each individual chapter.

As already mentioned, the early origins of computer audition techniques are in Speech recognition and processing [98]. Another important area of early influences is Computer

Music research [99, 85] which although is mostly involved with the synthesis of sounds and music has provided a variety of interesting tools and techniques that can be applied to Computer Audition.

Most of the algorithms in Music Information Retrieval (MIR) [4] has been developed based on symbolic representations such as MIDI files. Symbolic representations are similar to musical scores in that they are typically comprised of detailed structured information about the notes, pitches, timings, and instruments used in a recording. Based on this information MIR algorithms can perform various types of content analysis such as melody retrieval, key identification, and chord recognition. The techniques used have strong similarities to text information retrieval [137]. More information about symbolic MIR can be found in [1, 2].

Transcription of music is defined as the act of listening to a piece of music and writing down musical notation for the notes that constitute the piece. Automatic transcription refers to the process of converting an audio recording such as a .mp3 file to a structured representation such as a MIDI score and provides the bridge that can connect symbolic MIR with analysis of "real world" audio signals.

The automatic transcription of monophonic signals is practically a solved problem since several algorithms have been proposed that are reliable, commercially applicable and operate in real time. Attempts toward polyphonic transcription date back to the 1970s, when Moorer built a system for transcribing duets, i.e., two-voice compositions [86]. The proposed system suffered from severe limitations on the allowable frequency relations of two simultaneously playing notes, and on the range of notes in use. However, this was the first polyphonic transcription system, and several others were to follow.

Until these days transcription systems have not been able to solve other than very limited problems. It is only lately that proposed system can transcribe music 1) with more than two-voice polyphony, 2) with even a limited generality of sounds, and 3) yield

[4] `http://www.music-ir.org`

results that are mostly reliable and would work as a helping tool for a musician [79, 61]. Still, these two also suffer from substantial limitations. The first was simulated using only a short musical excerpt, and could yield good results only after a careful tuning of threshold parameters. The second system restricts the range of note pitches in simulations to less than 20 different notes. However, some flexibility in transcription has been attained, when compared to the previous systems. To summarize for practical purposes automatic polyphonic transcription of arbitrary musical purposes is still unsolved and little progress has been made.

Another approach that offers the possibility of automatic music analysis is Structured Audio (SA). The idea is to build a hybrid representation combining symbolic representation, software synthesis and raw audio in a structured way. Although promising this technique has only been used in a small scale and would require changes in the way music is produced and recorded. MPEG 4 [107] might change the situation but for now and for the near future most of the data will still be in raw audio format. Moreover, even if SA is widely used, converters from raw audio signals will still be required.

Given the fact that automatic music transcription is the holy grail of Computer Audition and is not very likely to be achieved in the near future and that Structured Audio is still mainly a proposal, if one wishes to work with audio signals today a significant paradigm shift is required. The main idea behind this paradigm shift is to try to extract directly information from audio signals using methods from Signal Processing and Machine Learning without attempting note-level transcriptions. This paradigm is followed in the work described in this thesis. Two position papers supporting this shift from note-level transcriptions to statistical approaches based on perceptual properties of ‘real-world” sounds even in the case when note-level transcriptions are available are [82, 106].

In my opinion, this statistical non-transcriptive approach to *Computer Audition* for non-speech audio signals really started in the second half on the 90s with the publications of two important influential papers. Statistical methods for the retrieval and classification of isolated sounds such as musical instrument tones and sound effects were introduced in [143]. An algorithm for the discrimination between music and speech using automatic classification based on spectral features and trained using supervised learning was described in [110]. An early overview of audio information retrieval including techniques based on Speech Recognition analysis can be found at [39].

Another influential work for CA is the classic book "Auditory Scene Analysis" by McGill Psychologist Albert Bregman [16]. Auditory scene analysis is the process by which the human auditory system builds mental descriptions of complex auditory environments by analyzing mixtures of sounds. The book describes a large number of psychological experiments that provide a lot of insight about the mechanisms used by the human auditory system to perform this process. Computational implementations of some of these ideas are covered in [101]. Although there is no direct attempt in this thesis to model the human auditory system, knowledge about how it works can provide valuable insight to the design of algorithms and systems that attempt to perform similar tasks.

More recently the Machine Listening Group of the MIT media lab has been active in CA producing a number of interesting PhD dissertations. The work of [31] is concerned with the problem of computational auditory scene analysis where an auditory scene is decomposed into a number of sound sources. The classification of musical instrument sounds is explored in [81]. The use information reduction techniques to provide mathematical descriptions of low level auditory processes such as preprocessing, grouping and scene analysis is described in [118]. The most relevant dissertation to this work is [109] which describes a series of systems for listening to music that do not attempt to perform polyphonic transcription.

1.8 Main Contributions

In this Section, the main new contributions of this thesis are briefly reviewed. More details and explanation of the terms will be provided in the following Chapters.

- **MPEG-based compressed domain spectral shape features**:

  Computation of features to represent the short-time spectral shape directly from MPEG audio compressed data (.mp3 files). That way the analysis done for compression can be used for "free" for feature extraction during decoding [130].

- **Representation of sound "texture" using texture windows**

  Unlike simple monophonic sounds such as musical instrument tones or sound effects music is a complex polyphonic signal that does not have a steady state and therefore has to be characterized by its overall "texture" characteristics rather than a steady state. The use of texture windows as a way to represent sound "texture" has been formalized and experimentally validated in [133].

- **Wavelet Transform Features**

  The Wavelet Transform (WT) is a technique for analyzing signals. It was developed as an alternative to the short time Fourier transform (STFT) to overcome problems related to its frequency and time resolution properties. A new set of features based on the WT that can be used for representing sound "texture" is proposed [134].

- **Beat Histograms and beat content features**

  The majority of existing work in audio information retrieval mainly utilizes features related to spectral characteristics. However if the signals of interest are musical signals more specific information such as the rhythmic content can be represented using features. Unlike typical automatic beat detection algorithms that provide only a running estimate of the main beat and its strength, Beat Histograms provide a richer representation of the rhythmic content of a musical piece. that can be used for similarity retrieval and classification. A method for their calculation based on periodicity analysis of multiple frequency bands is described [135, 133].

- **Pitch Histograms and pitch content features**

  Another important source of information specific to musical signals is pitch content. Pitch content has been used for retrieval of music in symbolic form where all the musical pitches the explicitly represented. However pitch information about audio signals is more difficult to acquire as automatic multiple pitch detection algorithm are still inaccurate. Pitch Histograms collect statistical information about the pitch content over the whole file and provide valuable information for similarity and classification even when the pitch detection algorithm is not perfect [133].

- **Multifeature segmentation methodology** Audio Segmententation is the process of detecting changes of sound "texture" in audio signals. A general multi-feature segmentation methodology that applies to arbitrary sound "textures" and does not rely on previous classification is described in this thesis. The proposed methodology can be applied to any type of audio features and has been evaluated for various feature sets with user experiments described in Chapter 5. This methodology was first described in [125] where segmentation as a separate process from classification and a more formal approach to multi-feature segmentation was provided. Additional segmentation experiments are described in [128].

- **Automatic musical genre classification**

  An algorithm for automatic musical genre classification of audio signals is described. It is based on a set of features for describing musical content that in addition to the typical timbral features, includes features for representing rhythmic and harmonic aspects of music. A variety of experiments have been conducted to e evaluate this algorithm and, to the best of our knowledge, this is the first detailed exploration of automatic musical genre classification. An initial version of the algorithm was published in [135] and the complete algorithm and more evaluation experiments are described in [133].

- **TimbreSpace** The *TimbreSpace* is a content and context aware visualization of audio collections where each audio signal is represented as a visual object in a 3D space. The shape, color and position of the objects are derived automatically based on automatic feature extraction and Principal Component Analysis (PCA).

- **TimbreGrams** *TimbreGrams* are content and context aware visualizations of audio signals that use colored stripes to show content structure and similarity. The coloring is derived automatically based on automatic feature extraction and Principal Component Analysis (PCA).

These short descriptions of each contribution are just indicative of the main ideas. More complete descriptions follow in the remaining Chapters.

1.9 A short tour

In this section a short tour of the proposed manipulation, analysis and retrieval systems is provided by describing some possible usage scenarios. The emphasis is on how the systems can be used rather than how they work. The interested reader might want to return to this section after reading the subsequent chapters that explain how the proposed systems work to see again how they fit in an application context. Some of the terms used might be unfamiliar (they will be explained in the rest of the thesis) but the gist of each scenario should be clear. These scenarios are indicative of the requirements of interacting with large audio collections and show how the systems we are going to describe meet these requirements.

A music listener is driving a car, listening to music through a computer audition enhanced radio receiver. The receiver is configured by the user to automatically change radio station when there are commercials and to automatically adjust equalizer setting based on the musical genre that is played (which is detected automatically). At some point, the listener hears a strange polyphonic vocal piece that is very interesting but does not know what it is. The piece is recorded and submitted to a music search engine. Several similar pieces are returned and the listener finds out that the piece is pygmy polyphonic music from Africa. The retrieved pieces can be subsequently by explored for example using a Soundslider to find a faster piece of similar texture.

A researcher in Bioacoustics receives a 24-hour long field recording of a site with birds and has to identify if any of the bird songs correspond to a list of endangered bird species. First the individual birdsongs are separated using the enhanced sound editor and automatic segmentation. Subsequently the researcher creates a collection consisting of the individual bird songs in addition to representative bird song samples from the endangered bird species list each labeled with a characteristic image. Finally the bird songs that are similar to the endangered species ones can be found by visual (they will be close to the target images) and aural (they will sound similar) inspection of an automatically calculated 2D or 3D timbrespace (a visual representation of large audio collections). By collecting a large number of representative birdsong samples an automatic classifier can be trained to recognize birdsongs of endangered species.

A musicologist is looking at the differences in duration of symphonic movements performed by different orchestras and conductors. Timbregrams (a visual representation of audio signals) of various symphonic works are calculated using a collection of orchestral music as the basis for Principal Component Analysis (PCA). That way each movement will be separated in color and movements of similar texture (for example loud, high energy movements) will have similar colors. The Timbregrams are then manually arranged in a table display such that each column corresponds to a symphonic work and each row corresponds to a specific conductor/orchestra combination. That way the differences in duration of each movement will be easily seen visually by comparing the lengths of similar colored movements across a specific column. By looking at a specific row and comparing it with the rest of the table the musicologist can draw conclusions such as conductor A tends to take faster high energy movements and slower low energy movements than conductor B.

A Jazz saxophone student wants to make a collection of saxophone solos by Charlie Parker. A TimbreSpace containing a collection of jazz music, where different jazz subgenres are automatically labeled by shape, is used initially. By selecting triangles corresponding to Bebop files and constraining the filenames to contain Charlie Parker a new collection of all the Bebop files with Charlie Parker is created. Using the enhanced sound editor with automatic segmentation and fine tuning the segmentation results manually all the saxophone solos are marked and saved as separate files. Finally a collection of saxophone solos is created from those saved files and can be used for subsequent browsing and exploration.

A sound designer wants to create the soundscape of walking through a forest using a database of sound effects. As a first step the sound of walking on soil has to be looked up. After clustering the sound effects collection the designer notices that a particular cluster has many files with Walking in their filename. Zooming on this cluster the designer selects 4 sound files labeled as walking on soil. Using spatialized audio generated thumbnails of the 4 sound files can be heard simultaneously in order to make the final decision. In a similar fashion files with bird songs can be located as well as the sound of a waterfall. Once the components of the soundscape are selected, they can be edit and mixed in order to create the final soundscape of walking through a forest. The illusion of approaching the waterfall can be achieved by artificial reverberation and crossfading.

It is evident from this scenarios that a large variety of algorithms for manipulating, analyzing and interacting with audio signals and collections are required. To make these scenarios even more interesting imagine that all these tasks are performed in front of a large screen with multiple graphic displays that are all consistent and connected to the same underlying data representation. Of course having such a large collaborative space allows more than one user to interact simultaneously with the system. The first steps toward such a system as well as variety of algorithms and interfaces that can used to realize these scenarios will be described in the following Chapters.

Chapter 2

Representation

If all things were at rest, and nothing moved, there would be perfect silence in the world. In such a state of absolute quiescence, nothing would be heard, for motion and percussion must precede sound. Just as the immediate cause of sound is some percussion, the immediate cause of all percussion must be motion. Some vibratory impulses or motions causing a percussion on the ear return with greater speed than others. Consequently they have a greater number of vibrations in a given time, while other are repeated slowly, and consequently are less frequent for a given length of time. The quick returns and greater number of such impulses produce the highest sounds, while the slower, which have fewer vibrations, produce the lower.

Euclid, Elements of Music, Introduction to the Section of the Canon

2.1 Introduction

The extraction of feature vectors as the main representation of audio signals, constitutes the foundation of most algorithms in *Computer Audition*. Feature extraction is the process of computing a numerical representation that can be used to characterize a segment of audio. This numerical representation, which is called the feature vector, is subsequently used as a fundamental building block of various types of audio analysis and information extraction algorithms. This vector has typically a fixed dimension and therefore can be thought of as a point in a multi-dimensional feature space. When using feature vectors to represent music two main approaches are used. In the first approach the audio file is broken into small segments in time and a feature vector is computed for each segment. The resulting representation is a time series of feature vectors which can be thought of as a trajectory of points in the feature space. In the second approach a single feature vector that summarizes information for the whole file is used. The single vector approach is appropriate when overall information about the whole file is required whereas the trajectory approach is appropriate when information needs to be updated in real time. For example, automatic musical genre classification of mp3 files might use the single vector approach while classification of radio signals might use the trajectory approach. Typically the signal is broken in small chunks called *analysis windows*. Their sizes are usually around 20 to 40 milliseconds. That way the signal characteristics are relatively stable for the duration of the window.

Most of the time, evaluation of audio features is done in the context of some specific application or analysis technique. So for example, a set of features for representing speech might be evaluated in the context of speech compression while a set of features for representing musical content might be evaluated in the context of automatic musical genre classification. Several cases of evaluation of the features described in this chapter are described in Chapter 5. The large variety of audio features that have been proposed differ not only in how effective they are but also in their performance requirements. For example, FFT-based features might be more appropriate for a real time application than a computationally intensive but perceptually more accurate features based on a cochlear filterbank.

The most relevant research area for audio feature extraction is Signal Processing. In many cases feature extraction is performed using some form of Time-Frequency analysis technique such as the Short Time Fourier Transform (STFT). Time-Frequency analysis technique basically represent the energy distribution of the signal in a time-frequency plane and differ in how this plane is subdivided into regions. The human ear also acts as a Time-Frequency analyzer decomposing the incoming pressure wave in the cochlea into different frequency bands.

2.2 Related Work

The origins of audio feature extraction are in the representation and processing of speech signals. A variety of representations for speech signals have been proposed in the literature. These representations where initially developed for the purposes of telephony and compression and later for speech recognition and synthesis. A complete overview of these techniques is beyond the scope of this Section. Interested readers can consult any of the standard textbooks on Speech processing such as [98]. Representations originating in speech processing that have been used in this thesis are: Linear Prediction Coefficients (LPC) [76] and Mel Frequency Cepstral Coefficients (MFCC) [26]. The use of MFCC for music/speech classification has been studied in [72]. These features will be explained in more detail later in this Chapter.

Toward the end of the 90s features for representing non-speech audio signals were proposed. Examples of non-speech audio signals that have been explored include: isolated instrument tones [44, 45, 32, 143], sound effects [143], and music/speech discrimination [110]. Another interesting direction of research during the 90s was the exploration of alternative front-ends to the typical Short Time Fourier Transform for the calculation of features. Representative examples are Cochlear Models [116, 117, 31, 109], Wavelets [65, 68, 122, 134], and the MPEG audio compression filterbank [95, 130].

Most of these papers utilize the proposed features for various audio analysis tasks such as classification or segmentation and will be described in more detail in the next Chapter. A short description of some representative examples is provided in this Section. Statistical moments of the magnitude spectrum are used as features for instrument classification in [44]. Spectral shape features such Centroid, Rolloff, and Flux, described later in this chapter, are used in [143] for sound effects and in [110] combined with MFCC-based features for music speech discrimination. Cochlear models attempt to approximate the exact mechanical workings of the lower levels of the human auditory system and have been shown to have similar properties for example in pitch detection [116]. Statistical characteristics of Wavelet subbands such as the absolute mean and variance of the coefficients have been used in [134] to model sound texture for the purposes of classification and segmentation. The use of the MPEG audio compression filterbank was independently proposed by [95] and [130] at the International Conference on Acoustics, Speech and Signal Processing (ICASP 2000). Recently systems that jointly process audio and video information from MPEG encoded sequences such as [14] have been proposed.

In the previously cited systems, the proposed acoustic features do not directly attempt to model musical signals and therefore ignore potentially useful information. For example no information regarding the rhythmic structure of the music is utilized. Research in the areas of automatic beat detection and multiple pitch analysis can provide ideas for the development of novel features specifically targeted to the analysis of music signals.

One characteristic of music that has received considerable attention is the automatic extraction of the beat or tempo of musical signals. Informally the beat can be defined as an underlying regular sequence of pulses that coincides with the flow of music. Of course this definition is vague but gives a general idea of the problem. A more anthropocentric way of defining beat is as the sequence of human foot tapping when one listens to music. More elaborate discussions about the definition of beat can be found in Musicology and Music

Cognition publications that are too numerous to be listed here. A good starting point is [67].

There is a significant body of work that deals with beat tracking and analysis using symbolic representations such as MIDI files. Although many interesting ideas have been proposed these techniques cannot deal with real audio signals. In contrast the systems that will be described in this Section are applicable to audio signals. A real-time beat tracking system for audio signals with music is described in [108]. In this system, a filterbank is coupled with a network of comb filters that track the signal periodicities to provide an estimate of the main beat and its strength. A real-time beat tracking system based on a multiple agent architecture that tracks several beat hypotheses in parallel is described in [49] [13]. A system for rhythm and periodicity detection in polyphonic music based on detecting beats at a narrow low frequency band (50-200 Hz) is described in [3]. A multiresolution time-frequency analysis and interpretation of musical rhythm based on Wavelets is described in [119]. More recently computationally simpler methods based on onset detection at specific frequencies have been proposed in [66, 112]. A beat tracking system for audio or MIDI data based on the frequency of occurrences of the various time durations between pairs of note onset times and multiple hypothesis search was introduced in [28] and enhanced with a user interface that provides visual and aural feedback in [29]. A beat extraction algorithm that operates directly on MPEG audio compressed bitstreams (.mp3 files) was recently proposed in [141].

The main output of these systems is a running estimate of the main beat and its strength. The beat spectrum, described in [43] is a more global representation of rhythm than just the main beat and its strength. Beat Histograms introduced in [134] and described in this chapter are a similar more global representation of rhythm (computed differently) that has been used for automatic musical genre classification [135, 133].

In addition to information related to the beat of music another important source of information is pitch content. Chord change detection (without chord name identification) has been used in [48], for real-time rhythm tracking for drumless audio signals. An algorithms capable of detecting multiple pitches in polyphonic audio signals is described in [123]. This algorithms has been used in [133] to calculate Pitch Histograms , a statistical representation of pitch content that is used for automatic musical genre classification.

The Short Time Fourier Transform is a standard Signal Processing technique and there are many references that describe it. A good introductory book is [121] and a more complete description can be found in [97, 89]. A good introduction to Wavelets, especially in relation to Signal Processing can be found in [77]. A high level overview of MPEG audio compression can be found at [88] and [58, 59] contain the complete specifications of the algorithm. For the the sake of completeness a short description of these techniques is provided in Appendix A for the interested reader.

2.2.1 Contributions

The following contributions of my this thesis to the area of audio feature extraction will be described in this chapter:

- **MPEG-based compressed domain spectral shape features**:

  Computation of features to represent the short-time spectral shape directly from MPEG audio compressed data (.mp3 files). That way the analysis done for compression can be used for “free” for feature extraction during decoding [130].

- **Representation of sound “texture” using texture windows**

  Unlike simple monophonic sounds such as musical instrument tones or sound effects music is a complex polyphonic signal that does not have a steady state and therefore has to be characterized by its overall “texture” characteristics rather than a steady state. The use of texture windows as a way to represent sound "texture" has been formalized and experimentally validated in [133].

- **Wavelet Transform Features**

  The Wavelet Transform (WT) is a technique for analyzing signals. It was developed as an alternative to the short time Fourier transform (STFT) to overcome problems related to its frequency and time resolution properties. A new set of features based on the WT that can be used for representing sound "texture" is proposed [134].

- **Beat Histograms and beat content features**

  The majority of existing work in audio information retrieval mainly utilizes features related to spectral characteristics. However if the signals of interest are musical signals more specific information such as the rhythmic content can be represented using features. Unlike typical automatic beat detection algorithms that provide only a running estimate of the main beat and its strength, Beat Histograms provide a richer representation of the rhythmic content of a musical piece. that can be used for similarity retrieval and classification. A method for their calculation based on periodicity analysis of multiple frequency bands is described [135, 133].

- **Pitch Histograms and pitch content features**

  Another important source of information specific to musical signals is pitch content. Pitch content has been used for retrieval of music in symbolic form where all the musical pitches the explicitly represented. However pitch information about audio signals is more difficult to acquire as automatic multiple pitch detection algorithm are still inaccurate. Pitch Histograms collect statistical information about the pitch content over the whole file and provide valuable information for similarity and classification even when the pitch detection algorithm is not perfect [133].

These short descriptions of each contributions are just indicative of the main ideas. More complete descriptions follow in the remaining Sections of this Chapter.

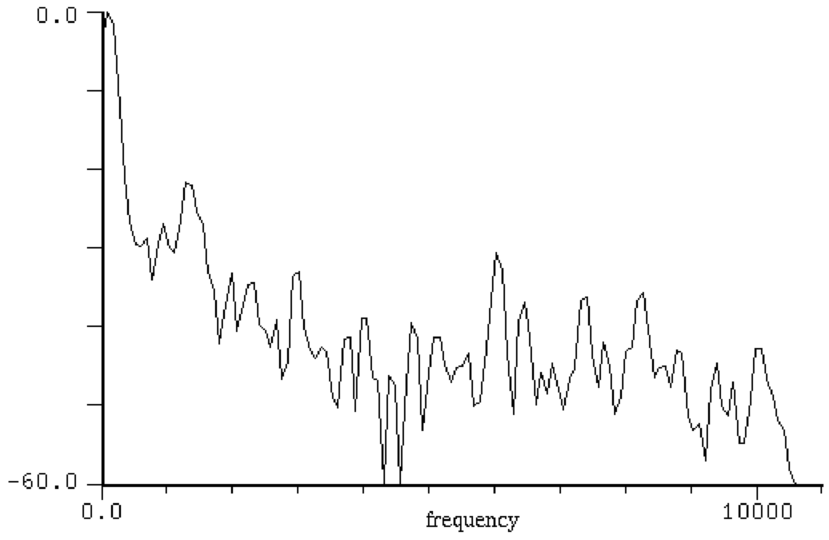


Figure 2.1: Magnitude Spectrum

2.3 Spectral Shape Features

Typically in Time-Frequency analysis techniques such as the STFT the signal is broken into small segments in time and the frequency content of each small segment is calculated. The frequency content can be represented as a magnitude spectrum (see Figure 2.1) that represents the energy distribution over frequency for that particular segment in time. The exact details of how the amplitudes are represented and how the frequency axis is spaced are important but for presentation purposes they will be ignored for now. Although the frequency spectrum can used directly to represent audio signals (in the extreme case can even be used to perfectly reconstruct them) it has the problem that it contains too much information. For CA purposes a more compact representation is more appropriate for two main reasons: 1) a lot of the information contained in the spectrum is not important 2) machine learning algorithms work better with feature vectors of small dimensionality that are as informative as possible. Therefore typically after the spectrum is calculated a small set of features that characterize the gross spectral shape are calculated. These types of features have been used in almost every type of Computer Audition and Speech Recognition algorithm. In the following subsections specific cases of features for characterizing the overall spectral shape are described.

2.3.1 STFT-based features

Features based on the Short Time Fourier Transform (STFT) are very common and have the advantage of fast calculation based on the Fast Fourier Transform algorithm. Although the exact details of the STFT parameters used to calculate them differ from system to system their basic description is the same. For the curious the following STFT analysis parameters have been used successfully in our implementation to represent musical and speech signals: sampling rate 22050, window size 512 (approximately 20 milliseconds), hop size 512 or 256, hanning window, amplitudes in dB. The following features are based on the short-time magnitude of the STFT transform of the signal:

Spectral Centroid

The spectral centroid is defined as the center of gravity of the magnitude spectrum of the STFT :

$$C_t = \frac{\sum_{n=1}^{N} M_t[n] * n}{\sum_{n=1}^{N} M_t[n]} \tag{2.1}$$

where $M_t[n]$ is the magnitude of the Fourier transform at frame t and frequency bin n. The centroid is a measure of spectral shape and higher centroid values correspond to “brighter” textures with more high frequencies. The spectral centroid has been shown by user experiments to be an important perceptual attribute in the characterization of musical instrument timbre [50].

Spectral Rolloff

The spectral rolloff is defined as the frequency R_t below which 85% of the magnitude distribution is concentrated:

$$\sum_{n=1}^{R_t} M_t[n] = 0.85 * \sum_{n=1}^{N} M_t[n] \tag{2.2}$$

The rolloff is another measure of spectral shape and shows how much of the signal's energy is concentrated in the lower frequencies.

Spectral Flux

The spectral flux is defined as the squared difference between the normalized magnitudes of successive spectral distributions:

$$F_t = \sum_{n=1}^{N} (N_t[n] - N_{t-1}[n])^2 \tag{2.3}$$

where $N_t[n]$, $N_{t-1}[n]$ are the normalized magnitude of the Fourier transform at the current frame t, and the previous frame $t-1$ respectively. The spectral flux is a measure of the amount of local spectral change. Spectral flux has also been shown by user experiments to be an important perceptual attribute in the characterization of musical instrument timbre [50].

2.3.2 Mel-Frequency Cepstral Coefficients

Mel-Frequency Cepstral Coefficients (MFCC) are perceptually motivated features that are also based on the STFT. After taking the log-amplitude of the magnitude spectrum, the FFT bins are grouped and smoothed according to the perceptually motivated Mel-frequency scaling. Finally, in order to decorrelate the resulting feature vectors a Discrete Cosine Transform is performed. Although typically 13 coefficients are used for Speech representation we have found that the first five coefficients are adequate for Music representation. A more detailed description of the MFCC calculation can be found in the Appendix.

2.3.3 MPEG-based features

Most of available audio data on the web is available in compressed form following the MPEG audio compression standard (the well known .mp3 files). MPEG audio compression is a lossy perceptually-based compression scheme that allows audio files to be stored in approximately one tenth of the space they would normally require if they were uncompressed. This enables large numbers of audio files to be stored and streamed over the network. Using current hardware the decompression can be performed in real-time and therefore in most cases there is no reason to store the uncompressed audio signal. Traditional Computer Audition algorithms operate on uncompressed audio signals therefore would require the signal to be decompressed and then subsequently analyzed. In MPEG audio compression the signal is analyzed in time and frequency for compression purposes. Therefore an interesting idea is to use this analysis not only for compression but also to extract features directly from compressed data. This is idea is similar to compressed domain visual processing of video (for example [146]). Because the bulk of the feature calculation is performed during the encoding stage this process has a significant performance advantages if the available data is compressed because the analysis is already there for "free". Combining decoding and analysis in one stage is also very important for audio streaming applications.

This idea of calculating features directly from MPEG audio compressed data was explored in [130] [1] and evaluated in the context of music/speech classification and segmentation. In this section the calculation of these features will be described and Chapter 5 describes how these features were evaluated. These features have been used in [71] for audio analysis of MPEG compressed video signals.

[1] a very similar idea about compressed domain audio features was independently proposed by David Pye at the same conference (ICASSP 2000) [95]

Short MPEG overview

The MPEG audio coding standard is an example of a perception based coder that exploits the characteristics of the human auditory system in order to compress audio more efficiently. Since there is no specific source model, like in speech compression algorithms, it can be used to compress any type of audio. In the first step the audio signal is converted to spectral components via an analysis filterbank. Each spectral component is quantized and coded with the goal of keeping the quantization noise below the masking threshold. Simultaneous masking is a frequency domain phenomenon where a low level signal (the maskee) can be made inaudible (masked) by a simultaneous occuring stronger signal (the masker) as long as masker and maskee are close enough to each other in frequency. Therefore, the dynamic bit allocation in each subband is controlled by a psychoacoustic model. Layer I,II,III offer different trade-offs between complexity and compression ratio. More details can be found in [88] and at the complete ISO specification [58, 59]. Essentially what MPEG audio compression does is assign quantization bits "cleverly" based on the properties of the human auditory system. [2]

Like most modern coding schemes there is an asymmetry between the encoder and the decoder. The encoder is more complicated, slower, and there is some flexibility in the psychoacoustic model used. On the other hand, decoding is simpler and straightforward. The subband sequences are reconstructed on the basis of blocks taking into account the bit allocation information. Each time the subband samples of all the 32 subbands have been calculated, they are applied to the synthesis filterbank, and 32 consecutive 16-bit, PCM-format audio samples are calculated. The feature calculation described in the following section is performed before the common to all layers filterbank synthesis.

The digital audio input is mapped into 32 subbands via equally spaced bandpass filters. A polyphase filter structure is used for the frequency mapping; its filters have 512 coef-

[2]Although there is no easy way to confirm this experimentally probably MPEG audio compression would not sound as good to non-human animals such as dogs that have different auditory characteristics

ficients. The filters are equally spaced in frequency. At a 22050 Hz sampling rate, each band has a width of 11025 / 32 = 345 Hz. The subsampled filter outputs exhibit significant overlap. The impulse response of subband k, $h_k(n)$ is obtained by multiplication of a single prototype low-pass filter, $h(n)$, by a modulating function that shifts the lowpass responce to the appropriate subband frequency range:

$$h_i(n) = h(n)cos(\frac{2i-1}{2M} + \phi(i));$$

$$M = 32; i = 1,2,..32; n = 1,2,...512; \tag{2.4}$$

Although the actual calculation of the samples is performed differently for performance reasons; the 32-dimensional subband vector can be written into a convolution equation:

$$S_t[i] = \sum_{n=0}^{511} x[t-n] * h_i[n] \tag{2.5}$$

where $h_i[n]$ are the individual subband band-pass filter responces. Details about the coefficients of the prototype filter and the phase shifts $\phi(k)$ are given in the ISO/MPEG standard [58], [59]. A more complete description of the MPEG audio compression standard can be found in Appendix A.

For the experiments MPEG-2, Layer III, 22050Hz sampling rate, mono, files were used. A similar approach can be used with the other layers and sampling rates as well as with any filterbank-based perceptual coder. The analysis is performed on blocks of 576 samples (about 20msec at 22050Hz) that correspond to one MPEG audio frame. A root mean squared subband vector is calculated for the frame as:

$$M[i] = \sqrt{\frac{\sum_{t=1}^{18}(S_t[i]^2)}{18}},\ i = 1..32; \tag{2.6}$$

S_t are the 32-dimensional subband vectors. The resulting M is a 32-dimensional vector that describes the spectral content of the sound for that frame. The characteristic features calculated are:

MPEG Centroid is the balancing point of the vector. It can be calculated using

$$C = \frac{\sum_{i=1}^{32} iM[i]}{\sum_{i=1}^{32} M[i]} \tag{2.7}$$

MPEG Rolloff is the value R such that

$$\sum_{i=1}^{R} M[i] = 0.85 \sum_{i=1}^{32} M[i] \tag{2.8}$$

MPEG Spectral Flux is the 2-norm of the difference between normalized M vectors evaluated at two successive frames.

MPEG RMS is a measure of the loudness of the frame. It can be calculated using

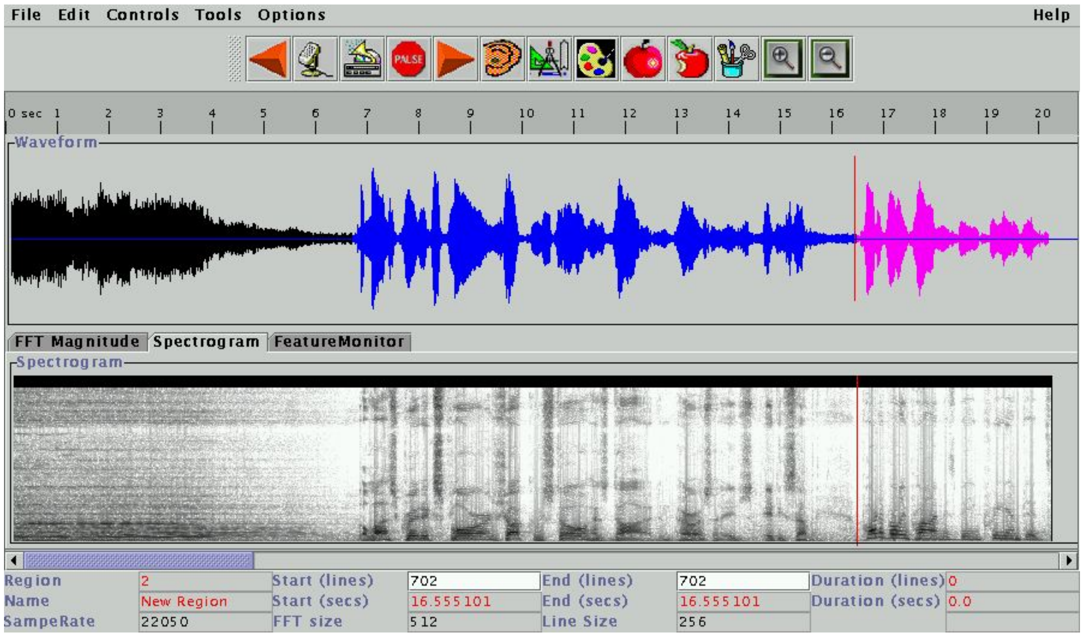
$$RMS = \sqrt{\frac{\sum_{i=1}^{32}(M[i]^2)}{32}} \tag{2.9}$$

This feature is unique to segmentation since changes in loudness are important cues for new sound events. In contrast, classification algorithms must usually be loudness invariant.

It is clear that these features are very similar to the features calculated on the magnitude spectrum of the STFT. This is not surprising as in both cases essentially what is summarized is the gross characteristics of the spectral shape. Comparisons of these features with other features such as STFT-based and MFCC in the content of audio classification and segmentation and will be provided in Chapter 5.

2.4 Representing sound “texture”

According to the standard definition the term “timbre” is used to describe all the characteristics of a sound that differentiate it from other sounds of the same pitch and loudness. This definition makes sense for single sound sources such as an isolated musical instrument tone or a single spoken vowel but can not be applied to more complex audio signals such as polyphonic music or speech. However, for different types of complex audio signals there are statistical qualities of the energy distribution in time and frequency that can be used to characterize them. For example we should expect heavy metal music to have quite different spectral characteristics than classical music, and speech is probably quite different from music. The term sound “texture” will be used to describe these statistical characteristics of the spectral distribution. It is important to note that these characteristics depend both on frequency and time. For example it is likely that heavy metal music will have more energy in the higher frequencies than classical music (frequency) and speech will have more changes of energy over time than most types of music.



Region	2	Start (lines)	702	End (lines)	702	Duration (lines)	0
Name	New Region	Start (secs)	16.555101	End (secs)	16.555101	Duration (secs)	0.0
SampeRate	22050	FFT size	512	Line Size	256		

Figure 2.2: Sound textures of news broadcast

Some examples of sound textures are: an orchestra playing, a guitar solo, a singing voice, a string quartet, a jazz big band, etc. Although in some cases these textures can be observed visually on a spectrogram, in most cases they can not. Figure 2.2 shows an excerpt from a news broadcast. There are three different textures (music, male speech, female speech), each one colored differently. It is easy to visually separate the music from speech from the spectrogram display but it is more challenging to separate male from female speech. So the goal is to somehow capture the statistical properties of sound "textures" using numerical features.

2.4.1 Analysis and texture windows

In short-time audio analysis the signal is broken into small, possibly overlapping, segments in time and each segment is processed separately. These segments are called *analysis windows* and have to be small enough so that the frequency characteristics of the magnitude spectrum are relatively stable i.e assume that the signal for that short amount of time is stationary. However, the sensation of a sound "texture" arises as the result of multiple short-time spectrums with different characteristics following some pattern in time. For example, speech contains vowel and consonant sections each of which have different spectral characteristics.

Therefore, in order to capture the long term nature of sound "texture", the actual features computed in our system are the running means and variances of the extracted features, described in the previous Sections, over a number of analysis windows. The term *texture window* will be used to describe this larger window and ideally should correspond to the minimum time amount of sound that is necessary to identify a particular sound or music "texture". Essentially rather than using the feature values directly, the parameters of a running multidimensional Gaussian distribution are estimated. More specifically, these parameters (means, variances) are calculated based on the *texture window* which consists of the current feature vector in addition to a specific number of feature vectors from the past. Another way to think of the *texture window* is as a memory of the past. For efficient implementation a circular buffer holding previous feature vectors can be used. In our system, an *analysis window* of 23 milliseconds (512 samples at 22050 Hz sampling rate) and a *texture window* of 1 second (43 analysis windows) is used. In Chapter 5 it will be shown experimentally that the use of “texture” windows improves the results of automatic musical genre classification.

2.4.2 Low Energy Feature

Low energy is the only feature that is based on the *texture window* rather than the *analysis window*. It is defined as the percentage of *analysis windows* that have less RMS energy than the average RMS energy across the *texture window*. For example musical signals will have lower “Low Energy” than speech signals that usually contain many silent windows.

2.5 Wavelet transform features

2.5.1 The Discrete Wavelet Transform

The Wavelet Transform (WT) is a technique for analyzing signals. It was developed as an alternative to the short time Fourier transform (STFT) to overcome problems related to its frequency and time resolution properties. More specifically, unlike the STFT which provides uniform time resolution for all frequencies, the WT provides high time resolution and low frequency resolution for high frequencies, and low time and high frequency resolution for low frequencies. In that respect it is similar to the human ear which exhibits similar time-frequency resolution characteristics.

The Discrete Wavelet Transform (DWT) is a special case of the WT that provides a compact representation of the signal in time and frequency that can be computed efficiently using a fast, pyramidal algorithm related to multirate filterbanks. More information about the WT and DWT can be found in Mallat [77]. For the purposes of this work, the DWT can be viewed as a computationally efficient way to calculate an octave decomposition of the signal in frequency. More specifically, the DWT can be viewed as a constant Q (center frequency / bandwidth) with octave spacing between the centers of the filters. A more detailed description of the DWT can be found in Appendix A.

2.5.2 Wavelet features

The extracted wavelet coefficients provide a compact representation that shows the energy of the signal in time and frequency. In order to further reduce the dimensionality of the extracted feature vectors, statistics over the set of wavelet coefficients are used. That way the statistical characteristics of the "texture" or "music surface" of the piece can be represented. For example the distribution of energy in time and frequency for music is different from that of speech.

The following following features are used to represent sound "texture":

- The mean of the absolute value of the coefficients in each subband. These features provide information about the frequency distribution of the audio signal.
- The standard deviation of the coefficients in each subband. These features provide information about the amount of change of the frequency distribution over time.
- Ratios of the mean absolute values between adjacent subbands. These features also provide information about the frequency distribution.

A window size of 65536 samples at 22050 Hz sampling rate with a hop size of 512 seconds is used a input to the feature extraction. This corresponds to approximately 3 seconds. Twelve levels (subbands) of coefficients are used resulting in a feature vector with 45 dimensions (12+12+11).

2.6 Other features

In this section some other supported features and do not fit in the previous categories will be briefly described.

RMS

RMS is a measure of the loudness of a window. It can be calculated using

$$RMS = \sqrt{\frac{\sum_{i=1}^{N}(M[i]^2)}{N}} \tag{2.10}$$

This feature is unique to segmentation since changes in loudness are important cues for new sound events. In contrast, classification algorithms must usually be loudness invariant.

Pitch

Pitch (as a audio feature) typically refers to the fundamental frequency of a monophonic sound signal and can be calculated using various different techniques [96].

Harmonicity

Harmonicity is measure of how strong the pitch detection for a sound is [143]. It can also be used for voiced/unvoiced detection.

Linear prediction (LPC) reflection coefficients

LPC coefficients are used in speech research as an estimate of the speech vocal tract filter [76].

Time domain Zero Crossings

$$Z_t = \frac{1}{2}\sum_{n=1}^{N}|sign(x[n]) - sign(x[n-1])| \tag{2.11}$$

where the *sign* function is 1 for positive arguments and 0 for negative arguments and $x[n]$ is the time domain signal for frame t. Time domain Zero Crossings provide a measure of the noisiness of the signal. For example heavy metal music due to guitar distortion and lots of drums will tend to have much higher zero crossing values than classical music. For clean (without noise) signals Zero Crossings are highly correlated with the Spectral Centroid. The evaluation of these features in the context of audio classification is described in Chapter 5.

2.7 Rhythmic Content Features

2.7.1 Beat Histograms

Most automatic beat detection systems provide a running estimate of the main beat and an estimate of its strength. In addition to these features in order to characterize musical genres more information about the rhythmic structure of a piece can be utilized. The regularity of the rhythm, the relation of the main beat to the subbeats, and the relative strength of subbeats to the main beat are some examples of characteristics of music we would like to represent through feature vectors.

A common automatic beat detector structure consists of signal decomposition into frequency bands using a filterbank, , followed by an envelope extraction step and finally a periodicity detection algorithm which is used to detect the lags at which the signal's envelope is most similar to itself. The process of automatic beat detection resembles pitch detection with larger periods (approximately 0.5 seconds to 1.5 seconds for beat compared to 2 milliseconds to 50 milliseconds for pitch).

The feature set for representing rhythm structure is based on detecting the most salient periodicities of the signal. Figure 2.3 shows the flow diagram of the beat analysis algorithm. The signal is first decomposed into a number of octave frequency bands using the

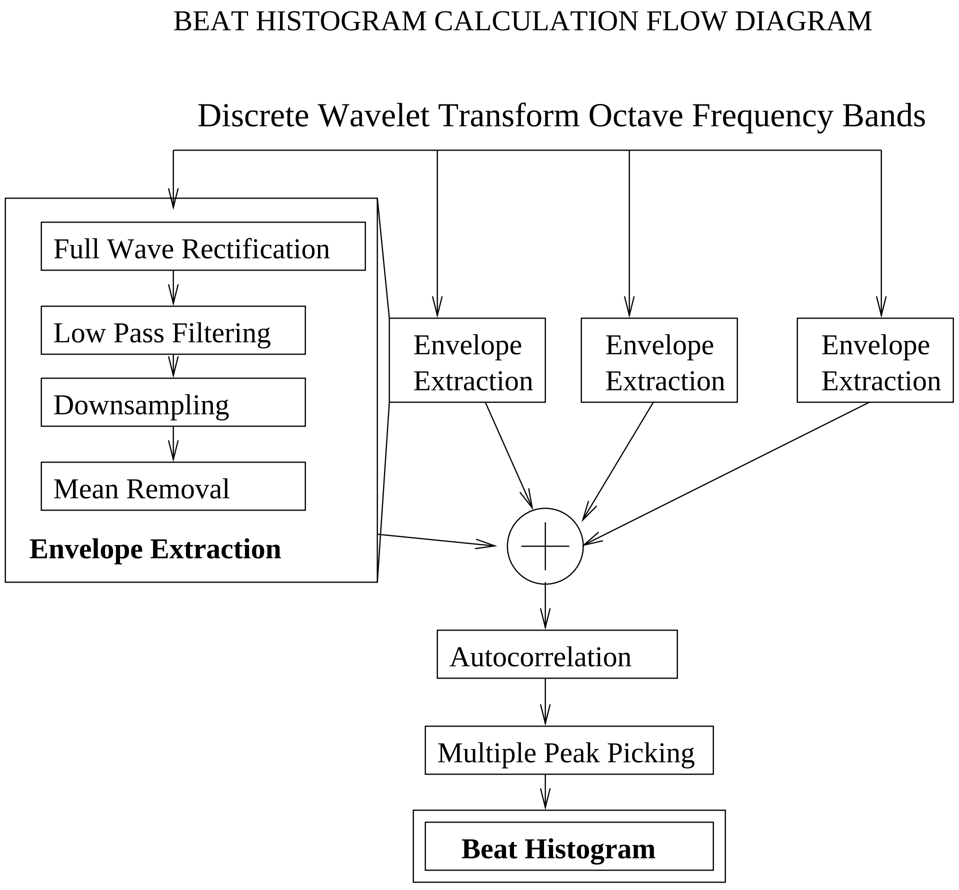


Figure 2.3: Beat Histogram Calculation Flow Diagram

DWT. Following this decomposition, the time domain amplitude envelope of each band is extracted separately. This is achieved by applying full-wave rectification, low pass filtering, and downsampling to each octave frequency band. After mean removal the envelopes of each band are then summed together and the autocorrelation of the resulting sum envelope is computed. The dominant peaks of the autocorrelation function correspond to the various periodicities of the signal's envelope. These peaks are accumulated over the whole sound file into a *Beat Histogram* where each bin corresponds to the peak lag (i.e the beat period in beats-per-minute bpm). Rather than adding one, the amplitude of each peak is added to the beat histogram. That way, when the signal is very similar to itself (strong beat) the histogram peaks will be higher.

The following building blocks are used for the beat analysis feature extraction:

Full Wave Rectification

$$y[n] = |x[n]| \tag{2.12}$$

is applied in order to extract the temporal envelope of the signal rather than the time domain signal itself.

Low Pass Filtering (LPF)

$$y[n] = (1-\alpha)x[n] + \alpha y[n-1] \tag{2.13}$$

i.e a One Pole Filter with an alpha value of 0.99 which is used to smooth the envelope. Full Wave Rectification followed by Low-Pass Filtering is a standard envelope extraction technique.

Downsampling

$$y[n] = x[kn] \tag{2.14}$$

where k=16 in our implementation. Because of the large periodicities for beat analysis, downsampling the signal reduces computation time for the autocorrelation computation without affecting the performance of the algorithm.

Mean Removal

$$y[n] = x[n] - E[x[n]] \tag{2.15}$$

is applied in order to make the signal centered to zero for the autocorrelation stage.

Enhanced Autocorrelation

$$y[k] = \frac{1}{N}\sum_{n} x[n]x[n-k] \tag{2.16}$$

The peaks of the autocorrelation function correspond to the time lags where the signal is most similar to itself. The time lags of peaks in the right time range for rhythm analysis correspond to beat periodicities. The autocorrelation function is enhanced using a similar method to the multipitch analysis model of [123] in order to reduce the effect of integer multiples of the basic periodicities. The original autocorrelation function of the summary of the envelopes, is clipped to positive values and then time-scaled by a factor of two and subtracted from the original clipped function. The same process is repeated with other integer factors such that repetitive peaks at integer multiples are removed.

Peak Detection and Histogram Calculation

The first three peaks of the enhanced autocorrelation function that are in the appropriate range for beat detection are selected and added to a *Beat Histogram (BH)*. The bins of the histogram correspond to beats-per-minute (bpm) from 40 to 200 bpm. For each peak of the enhanced autocorrelation function the peak amplitude is added to the histogram. That way peaks that have high amplitude (where the signal is highly similar) are weighted more strongly than weaker peaks in the histogram calculation.

2.7.2 Features

Figure 2.4 shows a beat histogram for a 30 second excerpt of the song “Come Together” by the Beatles. The two main peaks of the *Beat Histogram (BH)* correspond to the main beat at approximately 80 bpm and its first harmonic (twice the speed) at 160 bpm. Figure 2.5 shows four beat histograms of pieces from different musical genres. The upper left corner, labeled classical, is the BH of an excerpt from “La Mer” by Claude Debussy. Because of

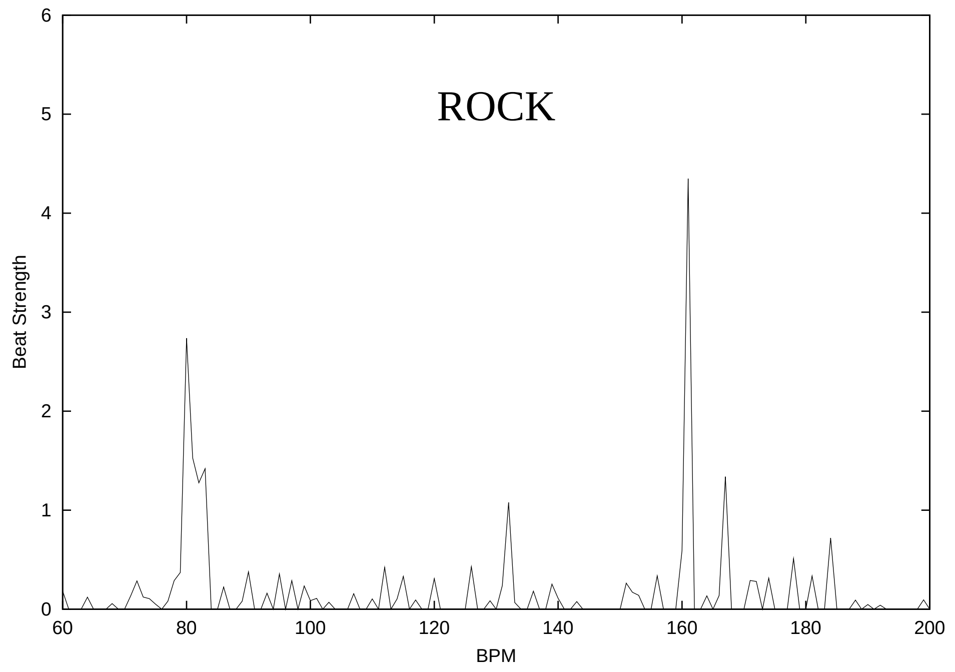


Figure 2.4: Beat Histogram Example

the complexity of the multiple instruments of the orchestra there is no strong self-similarity and there is no clear dominant peak in the histogram. More strong peaks can be seen at the lower left corner, labeled Jazz, which is an excerpt from a live performance by Dee Dee Bridgewater. The two peaks correspond to the beat of the song (70 and 140 bpm). The BH of Figure 2.4 is shown on the upper right corner where the peaks are more pronounced because of the stronger beat of rock music. The highest peaks of the lower right corner indicate the strong rhythmic structure of a HipHop song by Neneh Cherry.

Unlike previous work in automatic beat detection which typically aims to provide only an estimate of the main beat (or tempo) of the song and possibly a measure of its strength, the BH representation captures more detailed information about the rhythmic content of the piece that can be used to intelligently guess the musical genre of a song. Figure 2.5 indicates that the BH of different musical genres can be visually differentiated. Based on this observation a set of features based on the BH are calculated in order to represent

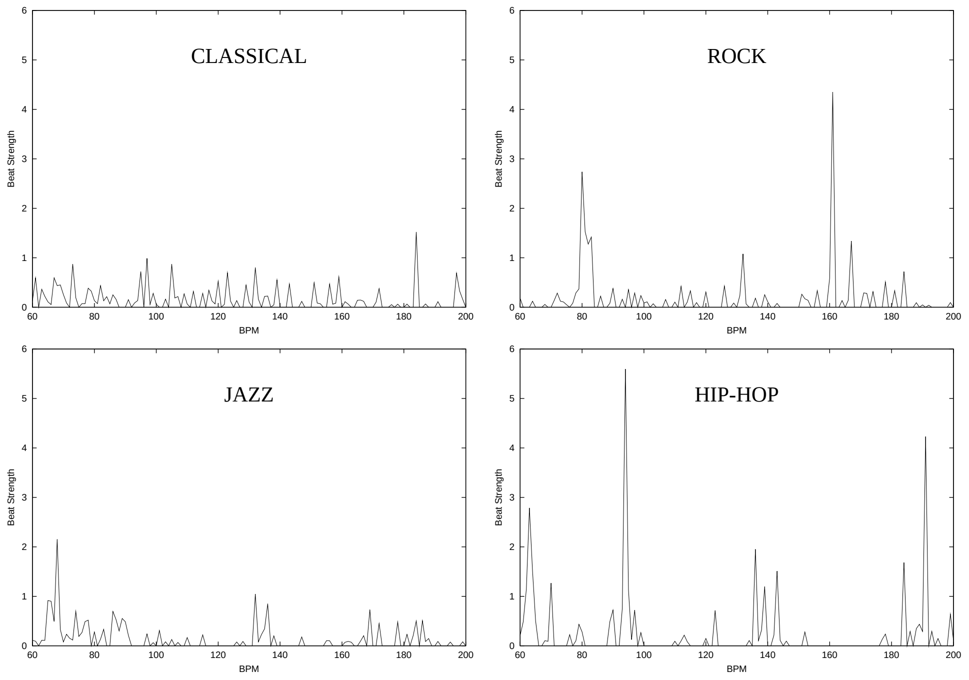


Figure 2.5: Beat Histogram Examples

rhythmic content and are shown to be useful for automatic musical genre classification. These are:

- **A0, A1**: relative amplitude (divided by the sum of amplitudes) of the first, and second histogram peak
- **RA**: ratio of the amplitude of the second peak divided by the amplitude of the first peak
- **P1, P2**: Period of the first, second peak in bpm
- **SUM**: overall sum of the histogram (indication of beat strength)

For the BH calculation, the DWT is applied in a window of 65536 samples at 22050 Hz sampling rate which corresponds to approximately 3 seconds. This window is advanced by a hop size of 32768 samples. This larger window is necessary to capture the signal repetitions at the beat and subbeat levels.

2.8 Pitch Content Features

2.8.1 Pitch Histograms

The pitch content feature set is based on multiple pitch detection techniques. More specifically, the multipitch detection algorithm described by [123] is utilized. In this algorithm, the signal is decomposed into two frequency bands (below and above 1000 Hz) and amplitude envelopes are extracted for each frequency band. The envelope extraction is performed by applying half-wave rectification and low-pass filtering. The envelopes are summed and an enhanced autocorrelation function is computed so that the effect of integer multiples of the peak frequencies to multiple pitch detection is reduced. More details about this algorithm can be found in the Appendix.

The prominent peaks of this summary enhanced autocorrelation function (SACF) correspond to the main pitches for that short segment of sound. This method is similar to the beat detection structure for the shorter periods corresponding to pitch perception. The three dominant peaks of the (SACF) are accumulated into a Pitch Histogram (PH) over the whole soundfile. For the computation of the PH, a pitch analysis window of 512 samples at 22050 Hz sampling rate (approximately 23 milliseconds) is used.

The frequencies corresponding to each histogram peak are converted to musical pitches such that each bin of the PH corresponds to a musical note with a specific pitch (for example A4 = 440 Hz). The musical notes are labeled using the MIDI note numbering scheme. The conversion from frequency to MIDI note number can be performed using the following equation:

$$n = 12 \log_2 \frac{f}{440} + 69 \tag{2.17}$$

where f is the frequency in Hertz and n is the histogram bin (MIDI note number).

Two versions of the PH are created: a *folded* (FPH) and *unfolded* histogram (UPH). The unfolded version is created using the above equation without any further modifications. In the folded case all notes are mapped to a single octave using the equation:

$$c = n \bmod 12 \tag{2.18}$$

where c is the folded histogram bin (pitch class or chroma value), and n is the unfolded histogram bin (or MIDI note number). The folded version contains information regarding the pitch classes or harmonic content of the music whereas the unfolded version contains information about the pitch range of the piece. The FPH is similar in concept to the chroma-based representations used in [10] for audio-thumbnailing. More information regarding the chroma and height dimension of musical pitch can be found in [113] and the relation of musical scales to frequency is discussed in more detail in [94].

Finally, the FPH is mapped to a circle of fifths histogram so that adjacent histogram bins are spaced a fifth apart rather than a semitone. This mapping is achieved by the following equation:

$$c' = (7 \times c) \bmod 12 \tag{2.19}$$

where c' is the new folded histogram bin after the mapping and c is the original folded histogram bin. The number 7 corresponds to 7 semitones or the music interval of a fifth. That way, the distances between adjacent bins after the mapping are better suited for expressing tonal music relations (tonic-dominant) and the extracted features result in better classification accuracy.

Although musical genres by no means can be characterized fully by their pitch content there are certain tendencies that can lead to useful feature vectors. For example Jazz or Classical music tend to have a higher degree of pitch change than Rock or Pop music. As a consequence Pop or Rock music pitch histograms will have fewer and more pronounced peaks than the histograms of Jazz or Classical music.

2.8.2 Features

Based on these observations the following features are computed from the UPH and FPH in order to represent pitch content:

- **FA0** Amplitude of maximum peak of the folded histogram. This corresponds to the most dominant pitch class of the song. For tonal music this peak will typically correspond to the tonic or dominant chord. This peak will be higher for songs that do not have many harmonic changes.
- **UP0** Period of the maximum peak of the unfolded histogram. This corresponds to the octave range of the dominant musical pitch of the song.
- **FP0** Period of the maximum peak of the folded histogram. This corresponds to the main pitch class of the song.
- **IPO1** Pitch interval between the two most prominent peaks of the folded histogram. This corresponds to the main tonal interval relation. For pieces with simple harmonic structure this feature will have value 1 or -1 corresponding to fifth or fourth interval (tonic-dominant).
- **SUM** The overall sum of the histogram. This is feature is a measure of the strength of the pitch detection.

2.9 Musical Content Features

Based on the previously described features it is possible to design a feature representation that captures timbral, rhythmic and harmonic aspects of musical signals. More specifically the following 30-dimensional feature vector can be constructed:

- **Timbre - Sound Texture** Mean over the whole file of texture STFT-based features (means and variances of Centroid, Rolloff, Flux, ZeroCrossings over texture window) and Low Energy (9 dimensions, 23 milliseconds analysis window, 1 second texture window).
- **Timbre - Sound Texture** Means over the whole file of texture MFCCs (means and variances of the first five MFCC coefficients over the texture window) (not counting the DC term) (10 dimensions, 23 milliseconds analysis window, 1 second texture window)
- **Beat content - Rhythm** Rhythmic content features based on Beat Histograms (6 dimensions, 3 seconds analysis window, 1.5 seconds hop size)
- **Pitch content - Harmony** Pitch content features based on Pitch Histograms (5 dimensions, 23 milliseconds analysis window)

Although there are many possible variations, this feature set forms the basic representation used by most of the described algorithms in this thesis and has been shown to be effective in representing musical content.

2.10 Summary

Feature vectors are the fundamental building blocks of that underlie the majority of existing work in Computer Audition. A set of features based on the Short Time Fourier Tranform (arguably the most common feature front-end) were presented. A similar set of features that can be calculated directly from MPEG audio compressed data were described. In order to represent sound “texture” statistical information about the change of short time characteristics over larger time intervals is required. The use of “texture” windows allows the statistical combination of short time spectral shape features in order to represent of sound “texture”. A set of features that describe spectral shape and texture based on the Discrete Wavelet Transform was also described. In addition to timbral and textural information, musical signals can be characterized by characteristics of their rhythmic and pitch content. Beat and Pitch Histograms are such statistical characterization of musical signals that can be used to calculate features that represent musical content.

Chapter 3

Analysis

Why do we listen with greater pleasure to men singing music which we happen to know beforehand, than to music which we do not know ? Is it because, when we recognize the song, the meaning of the composer, like a man hitting the mark, is more evident ? This is pleasant to consider. Or is it because it is more pleasant to contemplate than to learn ? The reason is that in the one case it is the acquisition of knowledge, but in the other it is using it and a form of recognition. Moreover, what we are accustomed to is always more pleasant than the unfamiliar.

Aristotle, Problems, Book 19, No. 5

3.1 Introduction

In the previous chapter various different ways of representing audio signals in the form of feature vectors were described. In this chapter we will see how these representations can be used to analyze and somehow “understand” audio signals in various ways. After feature extraction an audio signal can be represented in two major ways: 1) as a time series of feature vectors (or a trajectory of points in the feature space) 2) or as single feature vector (or point in the feature space). Essentially analysis techniques try to extract content and context information about audio signals and collections by trying to learn and analyze the geometric structure of the feature space.

Extracting content and context information both in humans and machines is a task that involves memory and learning. Therefore ideas from *Machine Learning* and especially *Pattern Classification* are important for the development of the proposed algorithms. *Auditory Scene Analysis* is the process by which the auditory system builds mental descriptions of complex auditory environments by analyzing mixtures of sounds [16]. From an ecological viewpoint, we try to associate events with sounds in order to understand our environment. Classification (what made this sound?), segmentation (how long did it last?) and similarity (what else sounds like that) are fundamental processes of any type of auditory analysis. In this chapter algorithms for automatically performing these tasks will be described. Although there is no attempt to model directly the human auditory system, knowledge about how it works has provided valuable insight to the design of the described algorithms and systems. For example there is significant evidence that the decisions for sequential and simultaneous integrations of sounds are based on multiple cues. Similarly, multiple features and representations are used in the proposed algorithms.

3.2 Related Work

Audio analysis is based on *Machine Learning* (ML) and specifically *Pattern Classification* techniques. Representative textbooks describing these areas are [30, 105]. Although ML techniques such as Hidden Markov Modes (HMM) have been used for a long time in *Speech Recognition* [98] the have only recently been applied to other types of audio signals.

When *Speech Recognition* techniques started being accurate enough for practical use they were used in order to retrieve multimedia information. For example the Informedia project at Carnegie Mellon [53] contains a terabyte of audiovisual data. Indexing the archive is done using a combination of speech-recognition, image analysis and keyword searching techniques. Audio analysis and browsing tools can enhance such indexing techniques for multimedia data such as radio and television broadcasts, especially for the regions that do not contain speech.

More recently audio classification techniques that include non-speech signals have been proposed. Most of these systems target the classification of broadcast news and video in broad categories like music, speech and environmental sounds. The problem of discrimination between music and speech has received considerable attention from the early work of Saunders [103] where simple thresholding of the average zero-crossing rate and energy features is used, to the work of Scheirer and Slaney [110] where multiple features and statistical pattern recognition classifiers are carefully evaluated. A similar system is used in [102] to initially separate speech from music and then detect phonemes or notes accordingly. Audio signals are segmented and classified into “music”, “speech”, “laughter” and non-speech sounds using cepstral coefficients and a Hidden Markov Model (HMM) in [63]. An heuristic rule-based system for the segmentation and classification of audio signals from movies or TV programs based on the time-varying properties of simple features is proposed in [147]. Signals are classified into two broad groups of music and non-music which are further subdivided into (Music) Harmonic Environmental Sound, Pure Music, Song, Speech with Music, Environmental Sound with Music and (Non-music) Pure Speech and Non-Harmonic Environmental Sound. A similar audio classification and segmentation method (music, speech, environment sound and silence) based on Nearest Neighbor and Learning Vector Quantization is presented in [74]. A solution to the more difficult problem of locating singing voice segments in musical signals is described in [12]. In their system, the phoneme activation output of an automatic speech recognition system is used as the feature vector for classifying singing segments. A system for the separation of speech signals from complex auditory scenes is described in [90].

Another type of non-speech audio classification system involves isolated musical instrument sounds and sound effects. A retrieval-by-similarity system for isolated sounds (sound effects and instrument tones) has been developed at Muscle Fish LLC [143]. 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. The features used in their system are statistics (mean, variance, autocorrelation) over the whole sound file of short time features such as pitch, amplitude, brightness, and bandwidth. Using the same dataset various other retrieval and classification approaches have been proposed. The use of MFCC coefficients to construct a learning tree vector quantizer is proposed in [38]. Histograms of the relative frequencies of feature vectors in each quantization bin are subsequently used for retrieval. The same dataset is also used in [68] to evaluate a feature extraction and indexing scheme based on statistics of the Discrete Wavelet Transform (DWT) coefficients. In [70] the same dataset is used to compare various classification methods and feature sets and the use of the Nearest Feature Line pattern classification method is proposed.

Similarity retrieval of music signals based on content is explored in [127]. A more specific similarity retrieval method targeted towards identifying different performances of the same piece is described in [144, 145]. A similar system that retrieves large symphonic orchestral works based on the overall energy profile and dynamic programming techniques is described in [41].

A multi-feature classifier based on spectral moments for recognition of steady state instrument tones is was initially described in [44] and subsequently improved in [45]. Another paper exploring the recognition of musical instrument tones from monophonic recording is [17]. A more general framework for the same problem is described in [80, 81]. A detailed exploration and comparison of features for isolated musical instrument tones is provided in [32].

A method for the automatic hierarchical classification of musical genres was initially described in [135] and improved in [133]. A more limited classification system (three genres: Jazz, Classical, Rock) using ideas from texture modeling in the visual domain is described in [27]. The problem of artist detection using neural networks and support vectors trained using audio features is addressed in [142]. A related problem to similarity retrieval is audio identification by content or audio fingerprinting which is the use of audio content to directly identify pieces of music in large database [5].

Segmentation algorithms can be divided in algorithms that require prior classification and those who don't. Segmentation systems that require prior classification in many cases are based on Hidden Markov Models (HMM). Some Examples are: the segmentation of recorded meetings based on speaker changes and silences described in [63], and the evaluation of music segmentation using different feature front-ends (MFCC, LPC, Discrete Cepstrum) described in [8]. Another approach has been to detect changes by tracking various individual audio features and then using heuristics combine their results to create segmentations [147]. A segmentation method based on self-similarity is described in [42]. An interesting approach for segmentation based on the detection of relative silence is described in [93]. A comparison of model-based, metric-based and energy-based audio segmentation for broadcast new data is described in [62]. A multifeature approach that does not require prior classification and formalizes the integration of multiple features was proposed in [125] and further evaluated with user experiments in [128].

3.2.1 Contributions

The main contribution of this thesis to the analysis stage of Computer Audition, is a general multi-feature methodology for audio segmentation with arbitrary sound "textures" that does not rely on previous classification. The proposed methodology can be applied to any type of audio features and has been evaluated for various feature sets with user experiments described in Chapter 5. This methodology was first described in [125] where segmentation as a separate process from classification and a more formal approach to multi-feature segmentation was provided. Additional segmentation experiments are described in [128]. In addition, we have shown that effective musical genre classification and content-based similarity retrieval are possible using the musical content feature set that represents timbral, rhythmic and harmonic aspects of music. An initial version of the automatic musical genre classification algorithm was published in [135] and the complete algorithm and more evaluation experiments are described in [133]

3.3 Query-by-example content-based similarity retrieval

In most content-based multimedia information retrieval systems the main method of specifying a query is by example. In that paradigm the user provides a multimedia object (for example an image or a piece of music) as the query and the system returns a list of similar multimedia objects ranked by their similarity. Similarity retrieval can also be used for automatic playlist generation [4].

Query-by-example (QBE) music similarity retrieval is implemented based on the musical content features (Section 2.9) using the single feature vector representation for each file. Because the computed features represent various characteristics of the music such as its timbral texture, rhythmic and pitch content, feature vectors that are close together in feature space will also be perceptually similar. A naive implementation of similarity retrieval would use the Euclidean distance between feature vectors and would return similar audio files ranked by increasing distance (the closest one to the query feature vector is the first match). Because the features have different dynamic ranges and probably are correlated, a Mahalanobis distance is used in our system. The Mahalanobis distance [75] between two vectors is defined as:

$$M(x,y) = \sqrt{(x-y)^T \Sigma^{-1} (x-y)} \tag{3.1}$$

where Σ^{-1} is the inverse of the feature vector covariance matrix.

The feature covariance matrix Σ is defined as:

$$\Sigma = E[xx^T] \tag{3.2}$$

The Mahalanobis distance automatically accounts for the proper scaling of the coordinate axes and corrects the correlation between features. Evaluating similarity retrieval is difficult as it typically involves conducting user experiments. User experiments conducted to evaluate this technique are described in Chapter 5. A demonstration of the system can be found at: `http://soundlab.princeton.edu/demo.php`

3.4 Classification

Classification algorithms attempt to categorize observed patterns into groups of related patterns or classes. The classification or description scheme is usually based on the availability of a set of patterns that have already been classified or described. This set of patterns is termed the training set and the resulting learning strategy is characterized as supervised. Learning can also be unsupervised, in the sense that the system is not given an a priori labeling of patterns, instead it establishes the classes itself based on the statistical regularities of the patterns.

The classification or description scheme usually uses one of the following approaches: statistical (or decision theoretic), syntactic (or structural), or neural. Statistical pattern recognition is based on statistical characterizations of patterns, assuming that the patterns are generated by a probabilistic system. Structural pattern recognition is based on the structural interrelationships of features. Neural pattern recognition employs the neural computing paradigm that has emerged with neural networks.

A huge variety of classification systems have been proposed in the literature and there is no clear cut choice about which is best as they differ in many aspects such as: training time, amount of training data required, classification time, robustness, and generalization. The main idea behind most of these classification schemes is to "learn" the geometric structure of the training data in the feature space and use that to estimate a "model" that can generalize to new unclassified patterns. Parametric classifiers assume a functional form for the underlying feature probability distribution while non-parametric ones use directly the training set for classification. Even a brief presentation of all the proposed classification systems is beyond the scope of this thesis. More details can be found in standard textbook on Pattern Classification such as [30, 105].

3.4.1 Classifiers

In this section, the main classification algorithms that are supported in our system will briefly be described. The emphasis is on what model is used to represent the feature vectors and not how the model parameters can be estimated from training data. More details about the training of the described classifiers can be found in Appendix A. Although by no means the only possible choices, these classification algorithms have been effectively used to construct practical audio analysis algorithms that in many cases run in real-time.

The Gaussian (GMAP) clasifier assumes each class can be represented as a multi-dimensional Gaussian distribution in feature space. The parameters of the distribution (means and covariance matrix) are estimated using the labeled data of the training set. This classifier is typical of parametric statistical classifiers that assume a particular form for the underlying class probability density functions. This classifier is more appropriate when the feature distribution is unimodal. The locus of equal distance points from a particular Gaussian classifier is an ellipse which is axis-aligned if the covariance matrix is diagonal.

The Gaussian Mixture Model classifier (GMM) model each class as a fixed-size weighted mixture of Gaussian distributions. For example, GMM3 will be used to denote a model that has 3 mixture components. The GMM classifier is characterized by the mixture weights and the means and covariance matrices for each mixture component. For efficiency reason typically the covariance matrices are assumed to be diagonal. The GMM classifier is typically trained using the Expectation-Maximization (EM) algorithm which is an iterative optimization procedure. A tutorial for the EM algorithm can be found in [84].

Unlike parametric classifiers, the K-Nearest Neighbor (KNN) 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 the KNN classifier, the K nearest neighbors to the point to be classified are calculated and voting is used to determine the class. An interesting result about the nearest neighbor classifier is that no matter what classifier is used, we can never do better than to cut the error rate in half over the nearest-neighbor classifier, assuming the training and testing data represent the underlying feature space topology [30].

Artificial Neural Network (ANN) are multilayer architectures of strongly connected simple components that can be trained to perform function approximation, pattern association, and pattern classification. A standard technique for the training of ANNs is backpropagation which refers to the process by which derivatives of network error, with respect to network weights and biases can be computed. This process can be used with a number of different optimization strategies. The architecture of a multilayer network is not completely constrained by the problem to be solved. The number of inputs to the network is constrained by the problem, and the number of neurons in the output layer is constrained by the number of outputs required by the problem. However, the number of layers between network inputs and the output layer and the sizes of the layers are up to the designer. It has been shown that the two-layer sigmoid/linear network can represent any functional relationship between inputs and outputs if the sigmoid layer has enough neurons. A standard backpropagation neural network trained using a gradient-descent type of optimization has been used in our system as a pattern classifier. Although the achieved classification performance is comparable (not better) to the other classifiers, training time for large collections is too long for practical purposes and therefore the results in Chapter 5 will be based mainly statistical pattern recognition classifiers. There is a lot of potential in the use of ANN with architecture that are tailored to problems of Computer Auditition rather than used as a generic classification algorithm.

In all the previously described classifiers a labeled training set was used (supervised learning). The K-means algorithm is a well-known technique for unsupervised learning where no labels are provided and the system automatically forms clusters based solely on the structure of the training data. In the K-means algorithm each cluster is represented by its centroid and the clusters are iterativly improved. More details can be found in Appendix A. The same algorithm can also be used for Vector Quantization (VQ), a technique where each feature vector is coded as an integer index to a codebook of representative feature vectors corresponding to the means of the clusters.

Having different classifiers is important in real world applications, because in addition to classification accuracy, other factors such as training speed, classification speed and robustness are important. For example, the K-NN family of classifiers in its basic form is computationally intensive and requires significant storage for classification of large data sets. On the other hand, classification and training using the simple Gaussian (GMAP) classifier are fast and therefore it can be used for real-time applications.

3.4.2 Classification schemes

Based on these classification algorithms and the audio features described in chapter 2 various audio classification schemes are supported in our system. These schemes will be used in Chapter 5 to evaluate various features sets and algorithms. For the purposes of this Chapter each classification scheme will be briefly described and a representative classification accuracy will be given (more detailed descriptions and results as well as the specific details of the features and classifiers used will be provided in Chapter 5). There has been little prior work in the Musical, Jazz and Classical genres classification schemes.

Music vs Speech

Separating music from speech is one of the first audio classifications that have been explored. Detecting speech and music segments is important for automatic speech recognition systems especially when dealing with real world multimedia data and good results can be achieved as music and speech have quite different spectral characteristics. In our implementation classification accuracy of 89% is achieved.

3.4.3 Voices

In addition to separating music from speech another important classification is detecting the gender of a speaker. In this classification scheme audio signals are classified into three categories: male voice, female voice, and sports announcing. Sports announcing refers to any type of voice over a loud noisy background. In addition to being an important source of information by itself, speaker gender identification can be used to improve speech recognition performance. Classification accuracy of 73% is achieved in our system.

Musical Genres

Using this classification scheme audio signals are automatically classified into one of the following categories: Classical, Country, Disco, HipHop, Jazz, Rock, Blues, Reggae, Pop, Metal. Classification accuracy of 61% is achieved in our system.

Classical Genres

Classical music can further be subdivided into: Choir, Orchestra, Piano, String Quartet. These subgenres are more close to instrumentation categories. Classification accuracy of 88% is achieved in our system.

Jazz Genres

Jazz music can further be subdivided into: BigBand, Cool, Fusion, Piano, Quartet, Swing. These subgenres are more close to instrumentation categories. Classification accuracy of 68% is achieved in our system.

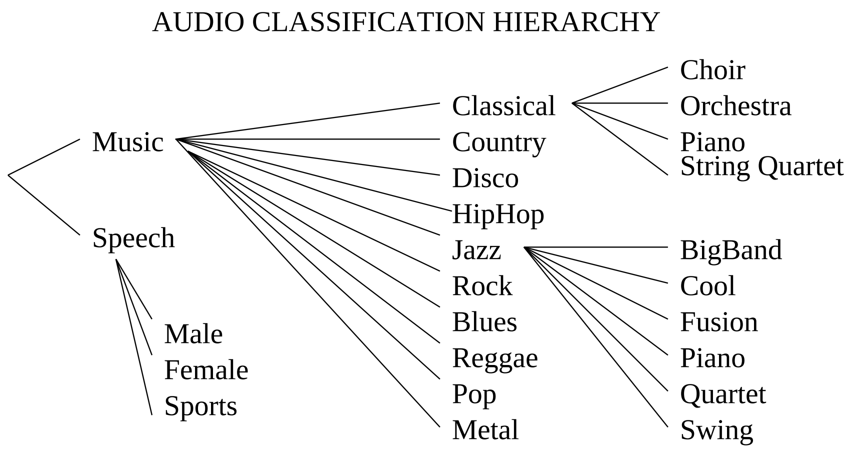


Figure 3.1: Musical Genres Hiearachy

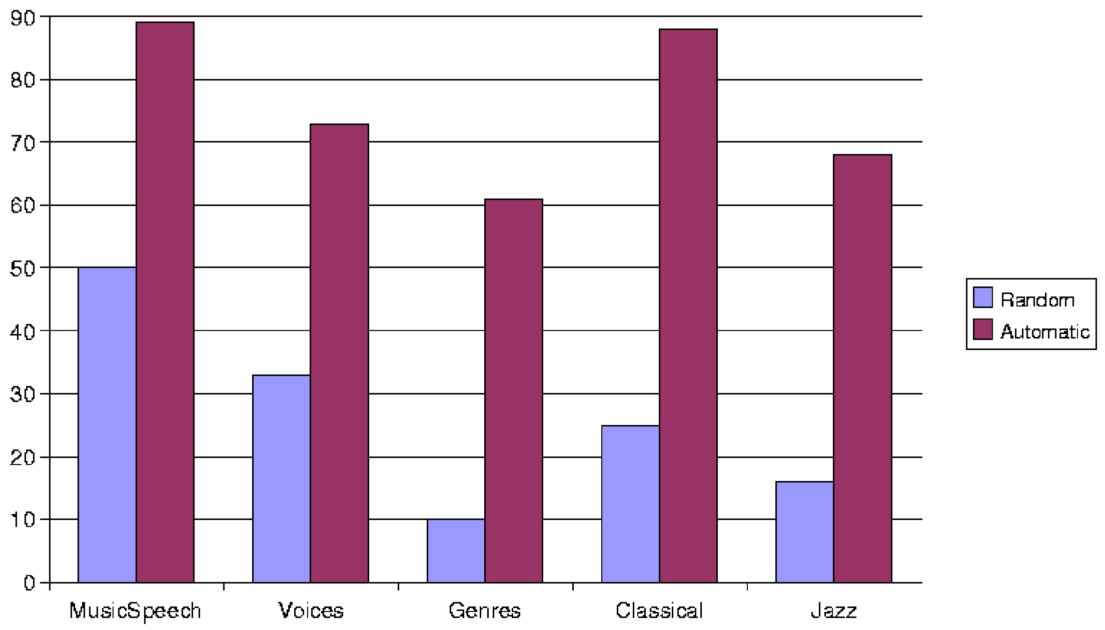


Figure 3.2: Classification Scemes Accuracy

Hierarchical Classification

These classification schemes are combined into an hierarchical schemes that can be used to classify radio and television broadcasts. Figure 3.1 shows this hierarchical scheme depicted as a tree with 23 nodes. A summary of the classification accuracy for each scheme is provided in Figure 3.2 and Table 3.1. There results are representative. More results and details about their calculation can be found in Chapter 5.

Other schemes

In addition to the classification schemes described above, other types of audio classification such as: clustering of sound effects, isolated instrument recognition, speaker emotion detection have been implemented using our system. Unlike the previously described clas-

Table 3.1: Classification schemes accuracy

	Random	Automatic
MusicSpeech(2)	50	89
Voices(3)	33	73
Genres(10)	10	61
Classical(4)	25	88
Jazz(6)	61	68

sification schemes, these types of classification are covered in the existing literature and will not be further described in this thesis. The results obtained are comparable with those reported in the published literature.

3.5 Segmentation

3.5.1 Motivation

One of the first chapters of most textbooks in image processing or computer vision is devoted to edge detection and object segmentation. This is because it is much easier to build classification and analysis algorithms using as input segmented objects rather than raw image data. In video analysis, shots, pans and generally temporal segments are detected and then analyzed for content. Similarly temporal segmentation can be used for audio and especially music analysis.

Auditory Scene Analysis is the process by which the human auditory system builds mental descriptions of complex auditory environments by analyzing mixtures of sounds [16]. From an ecological viewpoint, we try to associate events with sounds in order to understand our environment. The characteristics of sound sources tend to vary smoothly in time. Therefore abrupt changes usually indicate a new sound event. The decisions for sequential and simultaneous integration of sound are based on multiple cues. Although our method does not attempt to model the human auditory system, it does use significant changes of multiple features as segmentation boundaries. The experiments indicate that the selected features contain enough information to be useful for automatic segmentation.

Temporal segmentation is a more primitive process than classification since it does not try to interpret the data. Therefore, it can be more easily modeled using mathematical techniques. Being more simple it can work with arbitrary audio and does not pose specific constraints on its input like single speaker or isolated tones. It has been argued in [82] that music analysis systems should be built for and tested on real music and be based on perceptual properties rather than music theory and note-level transcriptions.

In this Section, a general segmentation methodology that can be used to segment audio based on arbitrary sound "texture" changes will be described. Some example of sound "texture changes" are a piano entrance after the orchestra in a concerto, a rock guitar solo entrance, a change of speaker etc.

Unlike many of the proposed algorithms for audio segmentation that rely on the existence of a prior classification, the proposed methodology is not constrained to fixed number of possible sound "textures". This makes applicable to a much broader class of segmentation problems. For example segmentation into music and speech regions can easily be handled by existing classification-based segmentation systems. However segmentation of different instrument sections of an orchestra using the classification-based is more difficult as a model for each possible subset of the orchestra should be learned. The proposed segmentation methodology has no problem handling such cases as it just detects the segmentation boundaries.

To further motivate segmentation suppose that a user is given the task of segmenting an Opera recording into sections and label each section appropriately with a text annotation. It can easily be verified experimentally that most of the time will be spend searching the audio for the section boundaries and much less time for the actual annotation. As an indication of the time required for such a task, 2 hours were the average time required by the subjects of experiments described in Chapter 5 to manually segment and annotate 10 minutes of audio using standard sound editing tools. These observations suggest a semi-automatic approach that combines manual and fully automatic annotation into a flexible, practical user interface for audio manipulation. Automatic segmentation is an important part of such a system. For example, the user can automatically segment audio into regions then run automatic classification algorithms that suggest annotations for each region. Then the annotations can be edited and/or expanded by the user. This way, significant amounts of user time are saved without loosing the flexibility of subjective annotation. Segmentation can also be used to detect musical structure of songs (for example cyclic ABAB type).

3.5.2 Methodology

The main idea behind the segmentation methodology, described in this Section, is that changes sound "texture" will correspond to abrupt changes in the trajectory of feature vectors representing the file. Although the statistics of a particular sound "texture" (and the corresponding feature vectors) might change over time significantly they will do this smoothly whereas if there is a "texture" change there will be an abrupt gap. The proposed methodology can utilize arbitrary sets of features (of course they have to be good representations of audio content but there is no constraint on their dimensionality and method of calculation). This is in contrast, to other proposed segmentation systems that are designed around specific features such as amplitude and pitch, track changes at each feature individually and use heuristics to combine the results.

The methodology can be broken into four stages:

1. A time series of feature vectors V_t is calculated by iterating over the sound file.

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

$$M(x,y) = (x-y)^T \Sigma^{-1} (x-y) \tag{3.3}$$

where Σ is the feature covariance matrix calculated from the whole sound file. This distance rotates and scales the feature space so the contribution of each feature is equal. 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. The derivative of the distance will be low for slowly changing textures and high during sudden transitions. The peaks roughly correspond to texture changes.

4. Peaks are picked using simple heuristics and are used to create the segmentation of the signal into time regions. As a heuristic example, adaptive thresholding can be used. A minimum duration between successive peaks can be set to avoid small regions.

For the experiments described in Chapter 5 the peak picking heuristic is parameterized by the desired number of peaks. More specifically the following heuristic is used:

1. The peak with the maximum amplitude is picked.

2. A region around and including the peak is zeroed (helps to avoid counting the same peak twice). The size of the region is proportional to the size of sound-file divided by the number of desired regions (20% of the average region size)

3. Step 1 is repeated until the desired number of peaks is reached.

One important parameter of the proposed segmentation methodology is the collection over which the feature covariance matrix Σ is calculated. The choice of collection makes the segmentation methodology context-dependent. For example the same file might be segmented differently if a collection of orchestral music is used for the computation of the covariance matrix compared to a collection of rock music.

3.5.3 Segmentation Features

The following set of features has been used successfully for automatic segmentation and are the basis of the experiments described in Chapter 5: Means and variances (over a texture window) of Spectral Centroid, Rolloff, Flux, ZeroCrossings, LowEnergy, and RMS. Loudness information expressed using the RMS feature is an important source of segmentation information that usually is ignored in classification-based segmentation algorithms as classification has to be loudness-invariant.

3.6 Audio thumbnailing

Audio Thumbnailing refers to the process of creating a short summary sound file from a large sound file in such a way that the summary best captures the essential elements of the original sound file. It is similar to the concept of key frames in video and thumbnails in images. Audio Thumbnailing is important for Audio MIR especially for the presentation of the returned ranked list since it allows users to quickly hear the results of their query and make their selection. In [73], two methods of audio thumbnailing are explored. The first is based on clustering and the second is based on the use of Hidden Markov Models (HMMs). According to the user experiments described in [73] both methods perform better than random selection. Clustering is the best method of the two. In addition to the clustering method, a segmentation-based method is supported in our system. In this method short segments around the segmentation boundaries are concatenated to form the summary. User experiments ([128]) have indicated that humans consider segmentation boundaries important for summarization.

3.7 Integration

Although the analysis algorithms were presented separately, their results can be integrated resulting in improved performance. As already mentioned in the related work section of this chapter, many proposed segmentation algorithms require prior segmentation. The other direction is also possible. 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 for example using a majority filter. 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 compares to fixed-window integration is planned for the future. Other examples of integration are the use of segmentation for thumbnailing, or the use of clustering in similarity retrieval to group the ranked list of similar files.

3.8 Summary

A number of fundamental audio analysis algorithms were presented in this chapter. The basic idea behind these algorithms is to learn and utilize the structure of the feature vector representation in order to extract information about audio signals. Figure 3.3 shows a high-level overview of the basic ideas described in this Chapter. The audio signal represented as a trajectory of feature vectors by breaking it into small *analysis windows* and computing

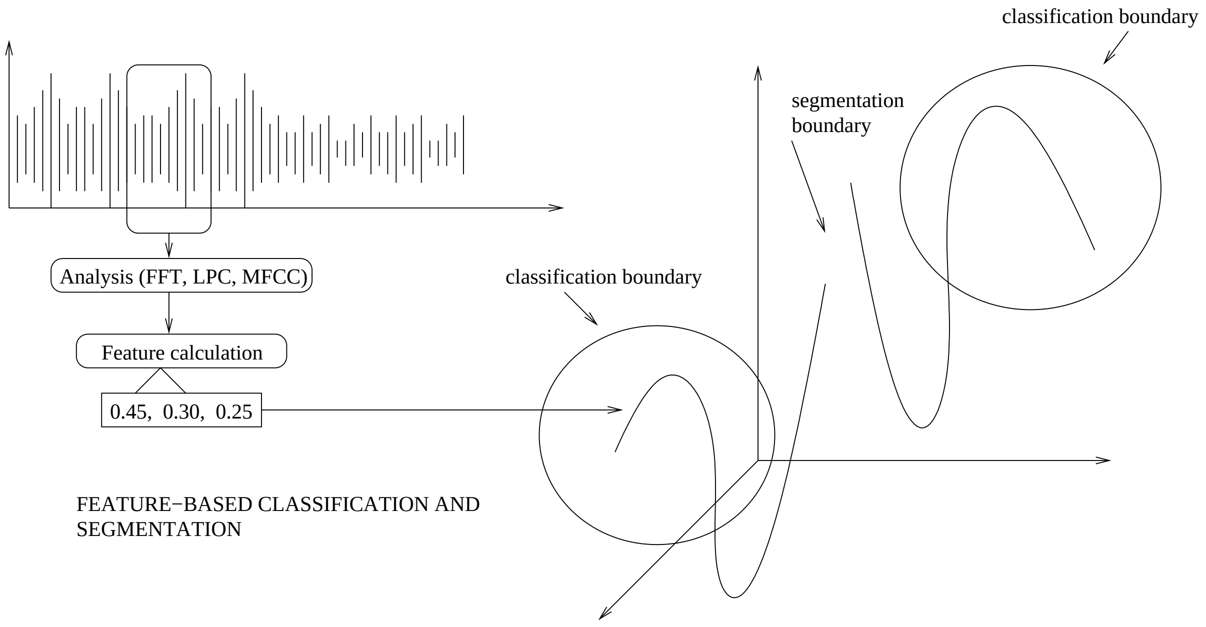


Figure 3.3: Feature-based segmentation and classification

a feature vector for each window. Segmentation can be performed by looking for abrupt changes in this trajectory of feature vectors and classification can be performed by partitioning the feature space in regions such that the points (vectors) falling in each partition belong to the same class.

More specifically the following analysis techniques were presented: content-based query-by-example similarity retrieval based on musical content features was described, hierarchical musical genre classification, a general multifeature segmentation methodology, and audio thumbnailing.

Chapter 4

Interaction

Thought is impossible without an image.

Aristotle, On the Soul, Book III

A Picture's Meaning Can Express Ten Thousand Words [1]

Phony Chinese Proverb

(and thousands of songs)

4.1 Introduction

The results of automatic audio analysis are still far from perfect and therefore need to be corrected, edited, and enhanced by a human user. Therefore the creation of user interfaces for interacting with audio signals and collections is an important direction for current research. Even if in the future automatic algorithms are vastly improved, music and sound

[1]Correct translation of phony chinese proverb typically translated as "A picture is worth ten thousand words". The "Chinese" quotation was fabricated by an advertising executive representing a baking soda company. The executive assumed that consumers would be compelled to buy a product that had the weight of Chinese philosophy behind it. A young boy's smile is equal to many words explaining the benefits of the product. The ad was often seen as a streetcar card ad that customers did not have much time to read. It appeared in Printers' Ink (now Marketing / Communication) March 10, 1927 (pp. 114-115)

listening is inherently subjective and therefore, in my opinion, the human user will always be part of the system.

Successful user interfaces combine the skills of humans and computers making effective use of their different abilities. User interfaces have been shown to provide significant improvements in almost every computer field ranging from the popular windows-based desktop to mission critical applications such as the design of NASA control rooms. Human Computer Interaction (HCI) studies the ways humans and computers interact and tries to improve them.

Computer audition research, in many cases originating from Electrical Engineering departments, has largely ignored user interface aspects. Although publications typically describe how the proposed algorithms can be used, they rarely show concrete examples of user interfaces that utilize their results. Such interfaces, in addition to demonstrating the proposed algorithms, are important for evaluation user experiments.

Currently the common way to interact with large collections of audio files is using commercial software mainly developed for audio recording and production purposes. Such sound editing software tools typically have well-designed and complex interfaces and allow the user to apply standard montage-like editing operations (cut, copy, paste, mix, etc), viewing operations (zoom in, zoom out, region select etc) and sound effects (filtering, denoising, pitch shifting etc) based on the traditional tape-recorder paradigm.

These current software tools for working with audio suffer from two major limitations. The first limitation is that they view audio as a monolithic block of digital samples without having any "understanding" of the specific audio content. For example, if users wish to locate the saxophone solo in a jazz piece, they have to manually locate the segment by listening, scrolling and zooming. Although Waveform and Spectogram displays can provide visual cues about audio content the process of locating regions of interest is still time consuming. The second limitation is that current audio software tools are centered around the concept of a file as the main unit of processing. Although typically multiple files can be opened and mixed, and batch operations can be applied to lists of files, there are no direct graphical user interfaces for displaying, browsing and editing large audio collections. Ideally these collection interfaces should also somehow reflect the content and similarity relations of their individual files.

In this Chapter, a series of different novel content and context aware interfaces for interacting with audio signals and collections are proposed and integrated in a common application for audio browsing, manipulation, analysis and retrieval. These components can utilize either manually or automatically extracted information or both. It is our belief, that this final case of utilizing both kinds of information provides the most effective way of interacting with large audio collections. The use of automatic techniques can significantly reduce the amount of user involvement and can provide detailed information while the human user can weed out the imperfections and make subjective judgments.

4.2 Related Work

Recently various graphical user interfaces for browsing image collections have been proposed spurred by the now commonplace availability of digital photography to computer users. Various spatial techniques and the use of thumbnail pictures have been proposed and shown improvements for the browsing and retrieval of images [60, 57, 100]. Several academic and commercial systems have been proposed for content-based retrieval of images. Some representative examples are the Blobword system [11] developed at UC Berkeley, the PhotoBook from MIT [91] and the QBIC system from IBM [37]. In these systems the user can search, browse and retrieve images based on similarity and various automatic feature extraction methods. This work falls under the general area of Multimedia Information Management Systems [51] with specific applications to audio signals.

In the context of audio signals, most of the existing work for browsing single files concentrates on the browsing of spoken documents. Speech skimmer [7] is a system for interactively skimming recorded speech that pushes audio interaction beyond the tape-recorder metaphor. The user can browse spoken documents by logical units such as words, sentences and changes of speaker. Time and Pitch modification techniques can be used to speed-up the speech signal without affecting intelligibility. Intelligent browsing of video signals of recorded meetings based on Hidden Markov Model (HMM) analysis using audio and visual features is described in [15]. Although there have been early attempts [18, 87] to develop “intelligent” sound editors most of their research effort was spend addressing problems related to the limited hardware of that time. Today, 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.

Many of the ideas described in this Chapter have their roots in the field of Scientific and Information Visualization [120, 34]. Visualization techniques have been used in many scientific domains. They take advantage of the strong pattern recognition abilities of the human visual system to reveal similarities, patterns and correlations in space and time. Visualization is more suited for areas that are exploratory in nature and where there are large amounts of data to be analyzed. Interacting with large audio collections is a good example of such an area. The concept of browsing is central to the design of the interfaces for interacting with audio collections described in this Chapter. Browsing is defined as “an exploratory, information seeking strategy that depends upon serendipity ... especially appropriate for ill-defined problems and exploring new task domains” [78]. The described components have been designed following Shneiderman’s mantra for the design of direct manipulation and interactive visualization interfaces: **“overview first, zoom and filter, then detail on demand”** [114].

Direct manipulation systems visualize objects and actions of interest, support rapid, reversible, incremental actions, and replace complex command-language syntax by direct manipulation of the object of interest [114]. Direct sonification refers to the immediate aural feedback to user actions. Examples of such direct manipulation systems include the popular desktop metaphor, computer-assisted-design tools, and video games. The main property of direct manipulation interfaces is the immediate aural and visual feedback in response to the user actions.

Of central importance to this work is the idea of representing sound as visual objects in a 2D or 3D space with properties related to the audio content and context. This idea has been used in psychoacoustics in order to construct perceptual spaces that visually show similarity relations between single notes of different musical instruments. These spaces can be constructed using Multidimensional Scaling (MDS) over data collected from user studies [50]. Another application that uses the idea of a space for audio browsing is the Intuitive Sound Editing Environment (ISEE) where 2D or 3D nested visual spaces are used to browse instrument sounds as experienced by musicians using MIDI synthesizers and samplers [138, 139].

The most related project to this work is the Sonic Browser which is a tool for accessing sounds or collections of sounds using sound spatialization and context-overview visualization techniques [35, 36]. In this work, a prototype 2D graphics browser combined with multi-stream sonic browsing was developed and evaluated. One application of this browsing system is musicological research in Irish traditional music. Although the goals of this work and the Sonic Browser are similar there are some difference in the two approaches. The main difference is that Sonic Browser emphasizes user interaction and direct manipulation without utilizing any automatically extracted information about the signals.

In Chapter 7 the combination of the Sonic Browser with our system to create a powerful sound browsing and editing environment will be described.

Visualizations of audio signals take advantage of the strong pattern recognition properties of the human visual system by mapping audio content and context information to visual representation. A visualization of audio signals based on self-similarity is proposed in [40]. Timbregrams, a content and context aware visualization for audio signals, were first introduced in [127]. Timbrespaces, a visualization for audio collection browsing, and the GenreGram monitor, a dynamic content-based real-time display, were introduced in [126]. A more complete description of the proposed interfaces with special emphasis on their use on the Princeton Display Wall is provided in [131].

4.2.1 Beyond the Query-by-Example paradigm

As we have seen, in recent years, techniques for audio and music information retrieval have started emerging as research prototypes. These systems can be classified into two major paradigms. In the first paradigm the user sings a melody and similar audio files containing that melody are retrieved. This approach is called "Query by Humming" (QBH) [47]. Unfortunately it has the disadvantage of being applicable only when the audio data is stored in symbolic form such as MIDI files. The conversion of generic audio signals to symbolic form, called polyphonic transcription, is still an open research problem in its infancy. Another problem with QBH is that it is not applicable to several musical genres such as Dance music where there is no singable melody that can be used as a query. In the second paradigm called "Query-by-Example" (QBE) [127] an audio file is used as the query and audio files that have similar content are returned ranked by their similarity. In order to search and retrieve general audio signals such as mp3 files on the web only the QBE paradigm is currently applicable.

In this Chapter new ways of browsing and retrieving audio and musical signals from large collections that go beyond the QBE paradigm will be proposed. The developed algorithms and tools rely on the automatic extraction of content information from audio signals. The main idea behind this work is to create new audio signals either by combination of other audio signals, or synthetically based on various constraints. These generated audio signals are subsequently used to auralize queries and parts of the browsing space. The ultimate goal of this work is to lay the foundations for the creation of a musical "sketchpad" which will allow computer users to "sketch" the music they want to hear. Ideally we would like the audio equivalent of sketching a green circle over a brown rectangle and retrieving images of trees (something which is supported in current content-based image retrieval systems).

In addition to going beyond the QBE paradigm for audio and music information retrieval this work differs from the majority of existing work in two ways: 1) continuous aural feedback 2) use of *Computer Audition* algorithms. Continuous aural feedback means that the user constantly hears audio or music that corresponds to her actions. *Computer Audition* techniques extract content information from audio signals that is used to configure the graphical user interfaces.

The term *Query User Interfaces* (QUI) will be used in this Chapter to describe any interface that can be used to specify in some way audio and musical aspects of the desired query. Two major families of *Query Interfaces* will be described based on the feedback they provide to the user. The first family consists of interfaces that utilize directly audio files in order to provide feedback while the second family consists of interfaces that generate symbolic information in MIDI format. It is important to note that in both of these cases the goal is to retrieve from general audio and music collections and not symbolic representations.

Another important influence for this work is the legacy of systems for automatic music generation and style modeling. These systems typically fall into four major categories:

generating music [13], assisting composition [83], modeling style, performance and/or composers [46], and automatic musical accompaniment [24]. These references are indicative and representative of early work in each catagory. There are also commercial systems that generate music according to a variety of musical styles such as the *Band in a box* software: `http:://www.sonicspot.com/bandinabox/bandinabox.html`

4.2.2 Contributions

Most the described content and context aware interfaces described in this Chapter are new although some extend or are based on older ideas. The most original contributions are the TimbreSpace browser and Timbregrams which are visualizations for audio collections and signals that express audio content and context information in the visual domain.

4.3 Terminology

The following terms will be used to describe the proposed systems:

Collection: a list of audio files (typical size $>$ 100 items)

Browser: an interface for interacting with a collection of audio files

Editor: an interface for interacting with a specific audio file

Monitor: a display that is updated in real-time in response to the audio that is being played

Mapping: a particular setting of interface parameters

Selection: a list of selected audio files in a collection or a specific region in time in an audio file

Viewer: a way to visualize a specific audio file

Timeline: a particular segmentation of an audio file to non-overlapping regions in time

View: a particular way of viewing a specific collection

The user interfaces described in this chapter follow a Model-View-Controller (MVC) framework [64]. The Model part comprises of the actual underlying data and the operations that can be performed to manipulate it. The View part describes specific ways the data model can be displayed and the Controller part describes how user input can be used to control the other two parts.

Collections are named lists of audio files. For example a *collection* might be named Rock 80s and consist of rock music files from that decade. A collection *selection* consists of an arbitrary subset of a collection. Typically a selection is specified by mouse operations. *Semantic zooming* refers to the process of zooming to a new collection consisting of only the selected objects. In the context of audio files a selection refers to a specific region in time of the audio file. A *timeline* is a segmentation of an audio file to non-overlapping possibly annotated regions in time. For example a jazz piece might have regions for the each individual instrument solo as well as the chorus. These concepts correspond to the Model part of the MVC framework.

The central idea behind the design of the described content and context aware GUIs is to map audio files to visual objects with specific properties related to their content using *viewers*. Collections of audio files are mapped to spatial arrangements of visual objects in *browsers* showing that way their context. A *mapping* describes a specific way the audio files are mapped to visual objects and their spatial arrangement. Multiple *views* of the same audio collection can be displayed possibly each with its own separate *mapping*. These concepts correspond to the View part of the MVC framework.

The controller part of the MVC framework in this work utilizes standard keyboard, mouse operations to control widgets such as buttons, sliders and scrollbars so there is no need to describe it in more detail. However, we note that separating this part from the Model and View components allows the easy incorporation of alternative methods of control such as speech input or virtual reality sensors to the system.

4.4 Browsers

4.4.1 Timbrespace Browser

The *TimbreSpace* browser maps each audio file to an object in a 2D or 3D space. The main browser properties that can be mapped are the x, y, z coordinates for each object. In addition a shape, texture image or color, and text annotation can be provided for each object. Standard graphical operations such as display zooming, panning, and rotating can be used to explore the browsing space. Data model operations such as selection pointing and semantic zooming are also supported. Selection specification can also be done by specifying constraints on the browser and object properties. For example the user can ask to select all the files that have positive x values, have triangular shape and have red color. Principal Curves originally proposed in [52] and used for sonification in [54], can be used to move sequentially through the objects.

TimbreSpaces can be constructed automatically based on the computation of audio features and the use of dimensionality reduction techniques such as Principal Components Analysis (PCA) [56]. PCA is described in more details in Appendix A. Dimensionality reduction techniques map a high dimensional set of feature vectors to a set of feature vectors of lower dimensionality with minimum loss of information. Using PCA and a single feature vector representation, each file is mapped to the x,y,z coordinates of a visual object. The mapping is achieved by mapping the first three principal components (which in many cases describe a most of the variability of the data collection) and mapping them to the unit cube. Another possibility for automatic configuration of TimbreSpace parameters is the direct use of feature values for each axis. For example, the x-axis might correspond to the estimated song tempo in bpms, the y axis to its length and the z-axis to auditory brightness. Finally, the object colors and/or shapes can be automatically configuration using classification and clustering algorithms.

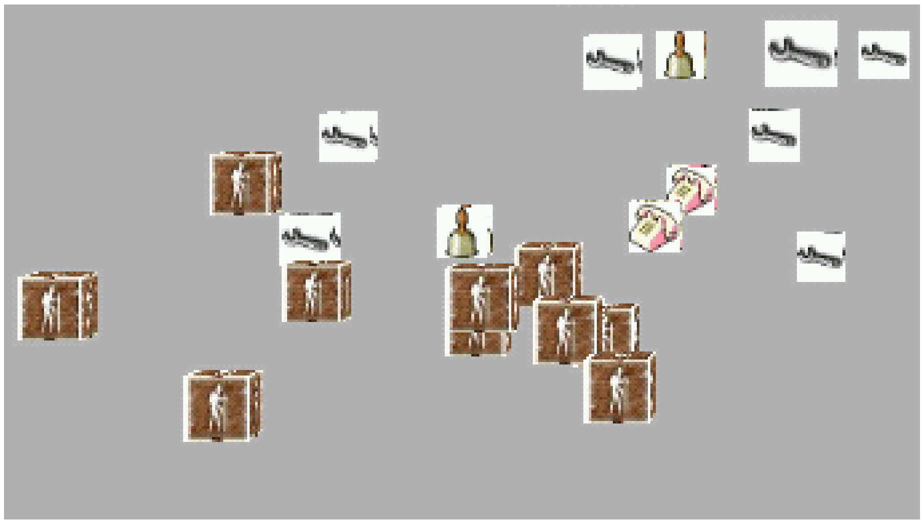


Figure 4.1: 2D TimbreSpace of Sound Effects

Figure 4.1 shows a 2D TimbreSpace of sound effects. The icons represent different types of sound effects such as walking (brown) and various other types of sound effects (white) such as tool, telephone and door bell sounds. Although the icons have been assigned manually the x,y coordinates of each icon are calculated automatically based on timbral texture features. That way files that are similar in content are visually clustered together as can be seen from the figure where the brown walking sounds occupy the left side of the figure while the white sounds occupy the right side.

Figure 4.2 shows a 3D TimbreSpace also shows a collection of soundeffects and similarly with Figure 4.1 the walking sounds are colored in dark. The walking sounds are automatically detected using K-Means clustering.

Figure 4.3 shows a 3D TimbreSpace of different pieces of orchestral music. Each piece is represented as a colored rectangle. The x,y,z coordinates are automatically extracted based on music similarity and the rectangle coloring is based on *Timbregrams* which are

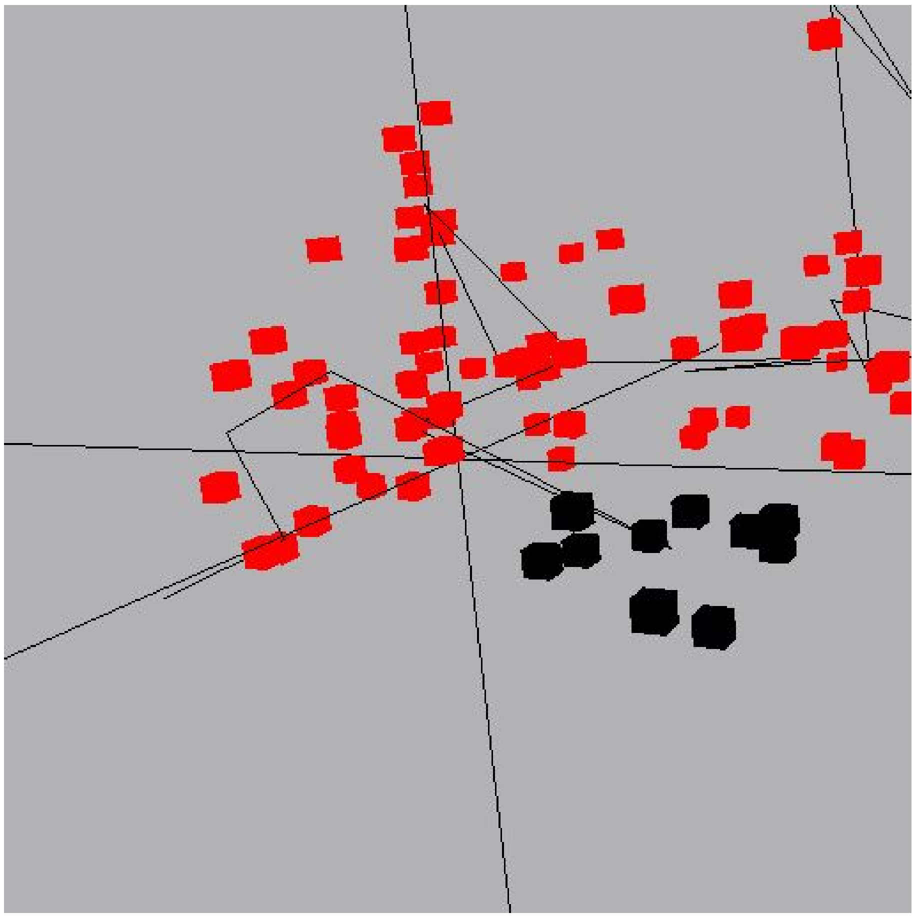


Figure 4.2: 3D TimbreSpace3D of Sound Effects

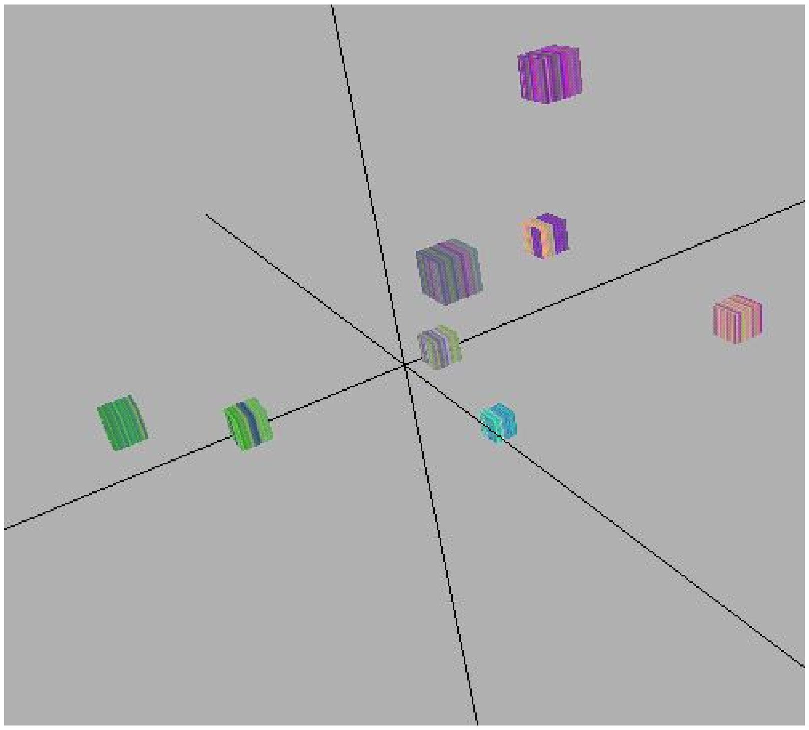


Figure 4.3: 3D TimbreSpace of orchestral music

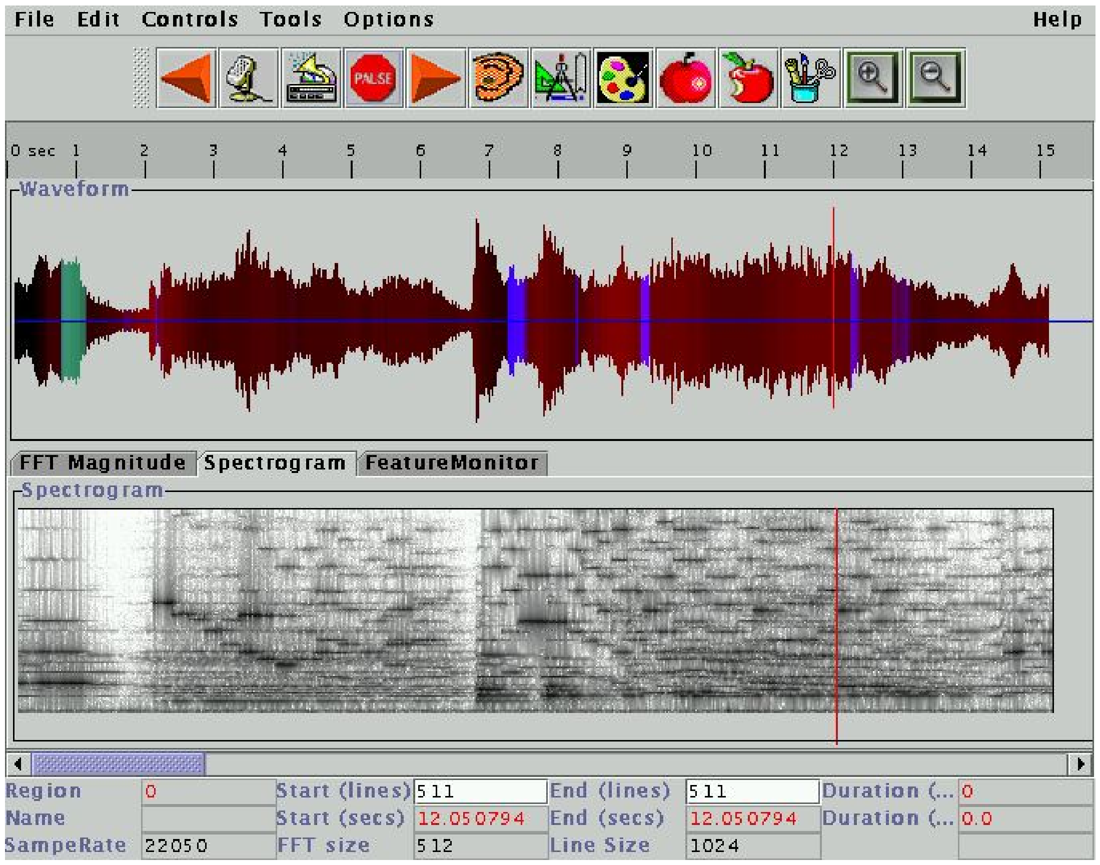
described in Section 4.4.2. Both figures contain fewer objects than typical configurations for clarity of presentation on paper. Audio collections have sizes typically ranging from 100 to 1000 files/objects.

4.4.2 Timbregram viewer and browser

The basic idea behind *TimbreGrams* is to map audio files to sequences of vertical color stripes where each stripe corresponds to a short slice of sound (typically sizes 20 milliseconds - 0.5 seconds). Time is mapped from left to right. The similarity of different files (context) is shown as overall color similarity while the similarity within a file (content) is shown by color similarity within the *Timbregram*. For example a file that has an ABA structure where section A and section B have different sound textures will have an ABA structure in color also. Although it is possible to manually created *Timbregrams* typically they are created automatically using Principal Components Analysis (PCA) over automatically extracted feature vectors.

Two main different approaches are used for mapping the principal components to color to create Timbregrams. If an indexed image is desired then the first principal component is divided equally and each interval is mapped to an index to a colomap. Any standard visualization colormap such as Greyscale or Thermometer can be used. This approach works especially well if the first principal component explains a large percentage of the variance of the data set. In the second approach the first three principal components are normalized so that they have equal means and variances. Although this normalization distorts the original feature space in practise it provides more satisfactory results as the colors are more clearly separated. Each of the first three principal components is mapped to coordinates in a RGB or HSV color space. In this approach a full color Timbregram is created. There is a tradeoff between the ability to show small scale local structure and global overall similarity depending on the quantization levels and the amount of variance allowed. For example by allowing many quantization levels and large variance different sections of the same audiofile will be visually separated. If only an overall color is desired fewer quantization steps and smaller variation should be used.

It is important to note that in this case, the similarity in color depends on the context i.e the collection over which PCA is calculated. That means that two files might have Timbregrams with similar colors as part of collection A and Timbregrams with different colors as part of collection B. For example a string quartet and orchestral piece will have different Timbregrams if viewed as part of a classical music collection but similar Timbregrams if viewed as part of a collection of arbitrary music. This is in contrast to techniques that directly map frequency or timbral content to color where the coloring of the file depends only on the file itself (content) and not the large collection it belongs to (context).



Region	0	Start (lines)	511	End (lines)	511	Duration (...	0
Name		Start (secs)	12.050794	End (secs)	12.050794	Duration (...	0.0
SampeRate	22050	FFT size	512	Line Size	1024		

Figure 4.4: Timbregram superimposed over waveform

Timbregrams can arranged in 2D tables for browsing. The table axis can be either computed automatically or manually created. For example one axis might correspond to the year of release and the other one might correspond to the tempo of the song. In addition *Timbregrams* can be superimposed over traditional Waveform displays (see Figure 4.4) and texture-mapped over objects in a *Timbrespace* (see Figure 4.3).

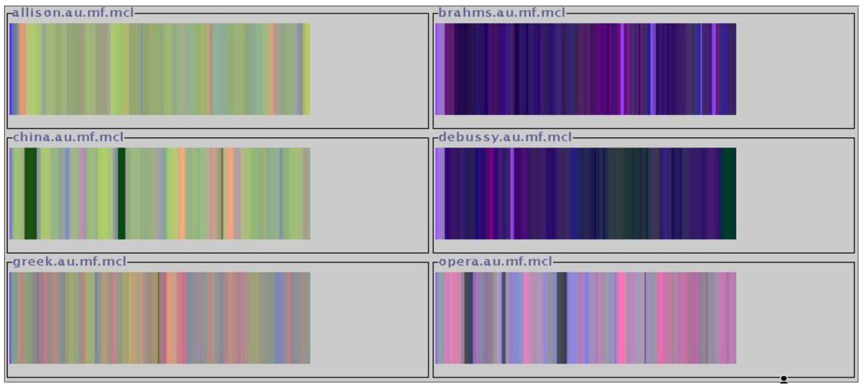


Figure 4.5: Timbregrams of music and speech

Figure 4.5 shows the *Timbregrams* of six sound files. The three files on the left column contain speech and the three on the right contain classical music. It is easy to visually separate music and speech even in the greyscale image. It should be noted that no explicit class model of music and speech is used and the different colors are a direct result of the visualization technique. The bottom right sound file (opera) is light purple and the speech segments are light green. In this mapping, light and bright colors correspond to speech or singining (Figure 4.5 left column). Purple and blue colors typically correspond to classical music (Figure 4.5. Similarly Figure 4.6 shows Timbregrams of Classical (left), Rock (middle), and Speech (right). Again the colors are a direct consequence of the visualization mapping and no class-model is used.

Timbregrams of pieces of orchestral music can be viewed in Figure 4.7. From inspecting visually the Figure it is clear that the fourth piece from the top has an AB structure where the A part is similar to the second piece from the top and the B part is similar to the

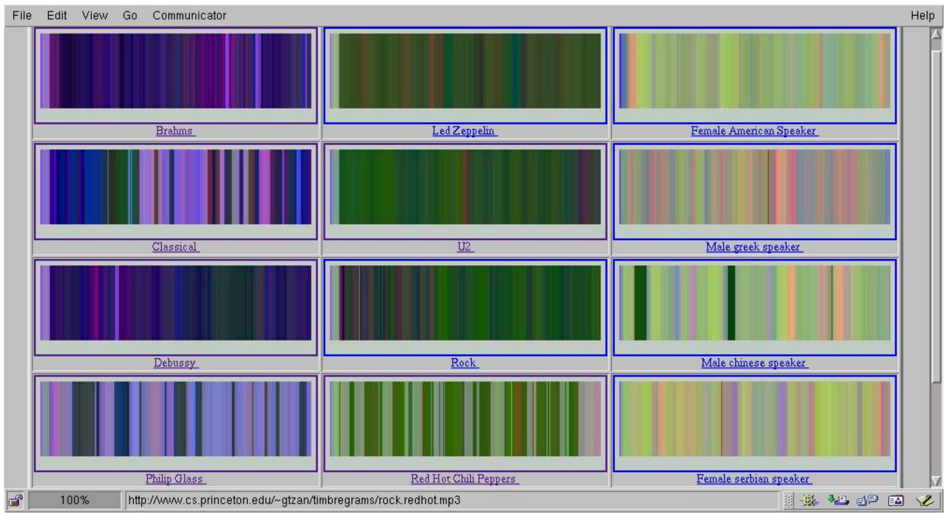


Figure 4.6: Timbregrams of classical, rock and speech

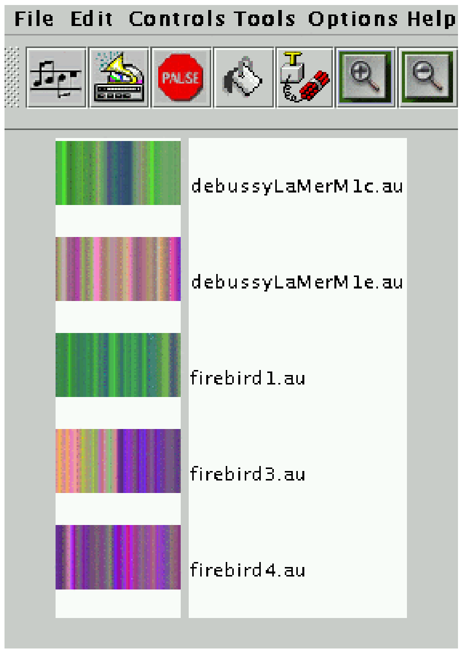


Figure 4.7: Timbregrams of orchestral music

last piece from the top. This is confirmed by listening to the corresponding pieces where A is a loud, energetic moment where all the orchestra is playing and B is a lightly orchestrated flute solo.

4.4.3 Other Browsers

The following browsers are based on existing graphical user interfaces and can be combined sharing the same underlying data representation in arbitrary ways with the previously described browsers. For example selecting a file in a TimbreSpace can also be reflected in a Tree browser and auralized using a SoundSpace.

Standard

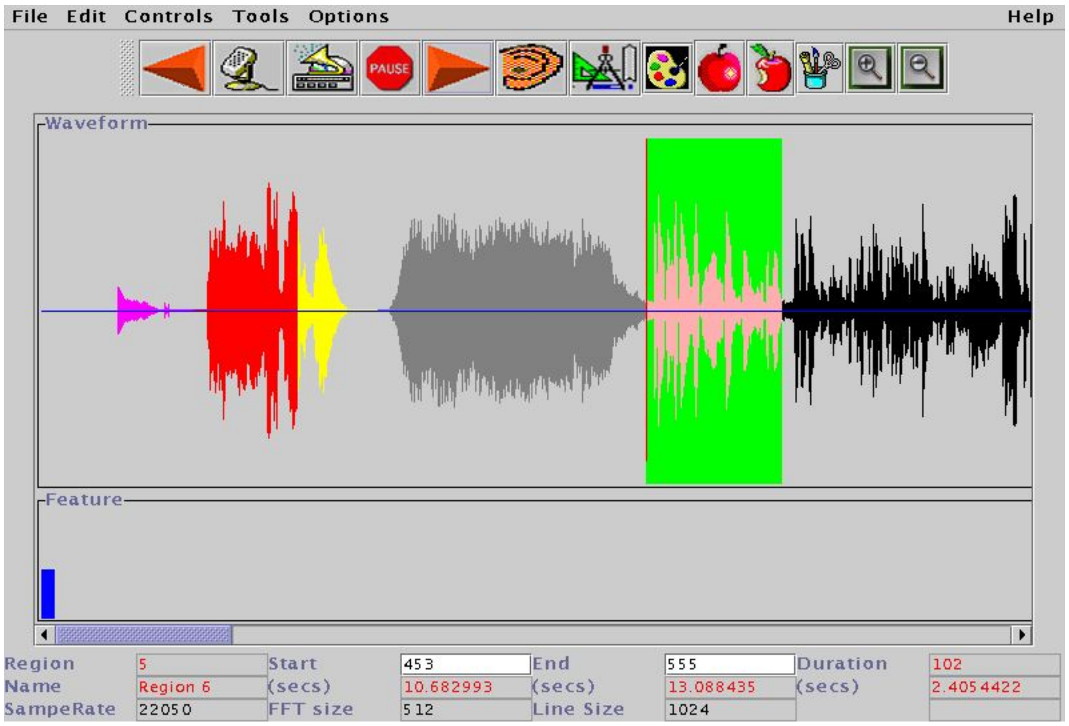
The standard browser provides a scrollable list of fileanames corresponding to the audio files of a collection. Standard single and multiple mouse selection are supported.

Tree

The Tree browser renders the audio signals of a collection based on a tree representation. This tree representation can be generated by automatic classification or created manually. For example the first level of the tree can be Music vs Speech and Speech can be further subdivided to Male vs Female and so forth.

SoundSpace

The SoundSpace browser takes advantage of the cocktail-party effect in a similar fashion to the Sonic Browser system described in [36]. As the user browses, a neighborhood (aura) around the currently selected item is auralized by playing all the corresponding files simultaneously. When used with the Princeton Display Wall (see Section 4.10 each file of the aura is mapped to one of the 16 loudspeakers of the system and the intensity levels



Region	5	Start	453	End	555	Duration	102
Name	Region 6	(secs)	10.682993	(secs)	13.088435	(secs)	2.4054422
SampeRate	22050	FFT size	512	Line Size	1024		

Figure 4.8: Enhanced Audio Editor I

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 audio thumnails can be automatically generated for each file.

4.5 Enhanced Audio Editor

The Enhanced Audio Editor 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 can either created by hand or using automatic audio segmentation.

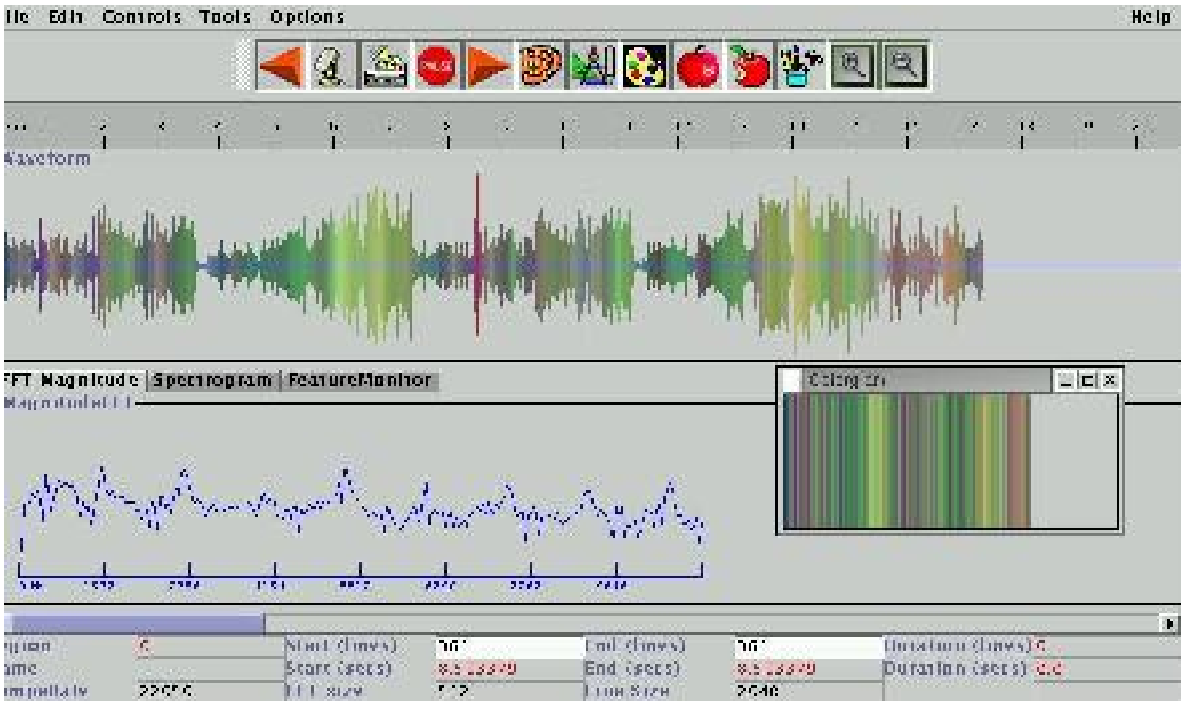


Figure 4.9: Enhanced Audio Editor II

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 schemes such as Music/Speech or Genre classification. 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.

Figure 4.8 shows an automatically created segmentation of an audio file. The user can browse by segments and automatically classify them. Figure 4.9 shows a snapshot of the editor containing a *Waveform display* with a *Timbregram* superimposed as well as a *Spectrum* display.

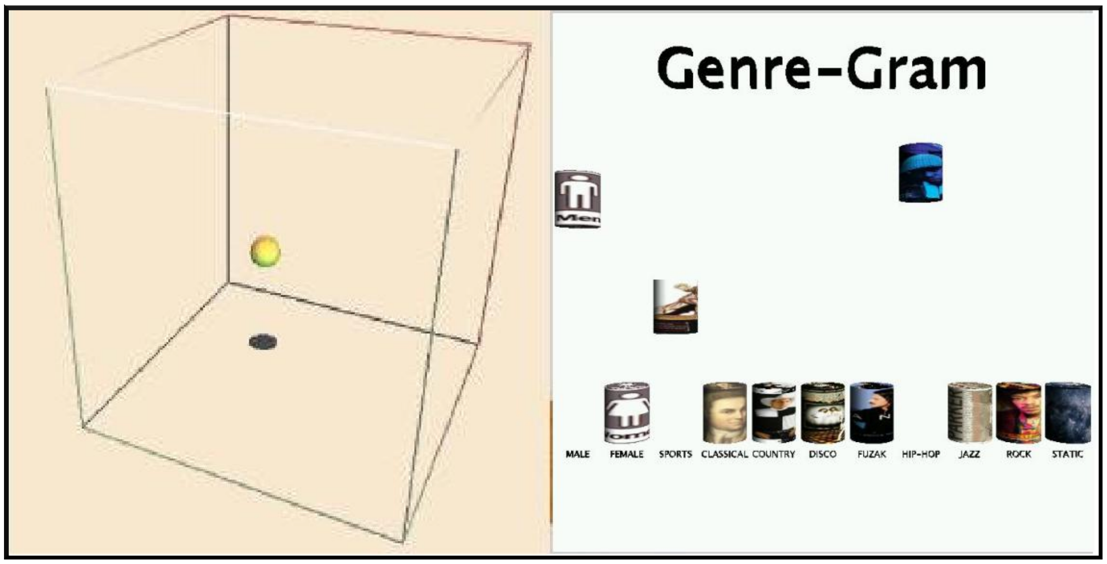


Figure 4.10: Timbreball and GenreGram monitors

4.6 Monitors

4.6.1 GenreGram

The *GenreGram* monitor is a dynamic real-time audio display for showing automatic genre classification results. Although it can be used with any audio signals it was designed for real-time classification of live radio signals. Each genre is represented as a cylinder that moves up and down in real-time based on a classification confidence measure ranging from 0.0 to 1.0. Each is cylinder is texture-mapped with a representative image for each genre. In addition to being a demonstration of real-time automatic musical genre classification the *GenreGram* provides valuable feedback both to users and algorithm designers. Different classifications decisions and their relative strengths are combined visually, revealing correlations and classification patterns. Since in many cases the boundaries between genres are fuzzy, a display like this is more informative than a single classificition decision. For example both male speech and hiphop are activated in the case of a hiphop song as shown in Figure 4.10. Of course it is possible to use *GenreGrams* to display other types of audio classification such as instrument, sound effects, and birdsong classification.

4.6.2 TimbreBall monitor

This monitor is a tool for visualizing in real-time the evolution of feature vectors extracted from an audio signal. In this animation each feature vector is mapped to the x,y,z coordinate of a small ball inside a cube. The ball moves in the space as the sound is playing following the evolution of the corresponding feature vectors. Texture changes are visible by abrupt jumps in the trajectory of the ball. A shadow is provided for better depth perception (see Figure 4.10.

4.6.3 Other monitors

These monitors are based on existing visualization techniques for audio signals.

Waveform

Display of signal's amplitude envelope familiar from commercial editors.

Spectrum

Display that shows the magnitude in decibels (dB) of the Short Time Fourier Transform (SFTT) of the signal. See Figure 4.9.

Spectogram

The 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. See Figure 4.4.

Waterfall

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

Metronome

This monitor shows the tempo of music in beats-per-minute (bpm) and a confidence level for how strong the beat is. The main beat and it's strength are detected automatically.

FeaturePlot

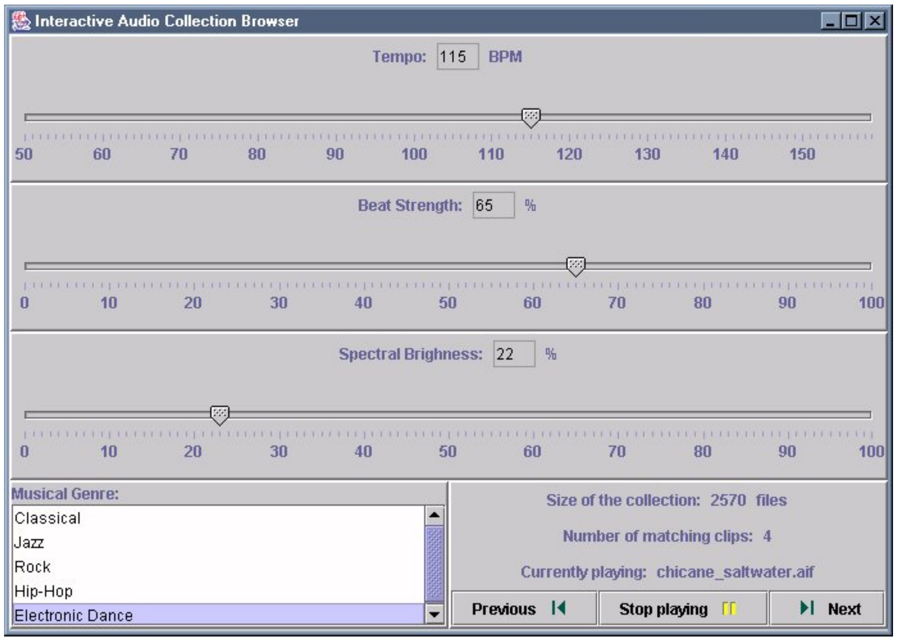
This monitor shows a plot of a single user-specified feature.

4.7 Audio-based Query Interfaces

In audio and music information retrieval there is a query and a collection that is searched for similar files. The main idea behind audio-based QUIs is to utilize not only the query's audio but also utilize directly the audio collection. Since the structure and content of the collection are already automatically or manually extracted for retrieval, this information can be used together with the audio to provide interactive aural feedback to the user. This idea will be clarified in the following sections with examples of audio-based QUIs.

4.7.1 Sound Sliders

Sound sliders are used to browse audio collections based continuous attributes. For presentation purposes assume that for each audio file in a collection the tempo and beat strength of the song are automatically extracted. For each of these two attributes a corresponding *sound slider* is created. For a particular setting of the sliders the sound file with attributes corresponding to the slider values is selected and part of it is played in a loop. If more than one sound file corresponds to the particular slider values, the user can advance in



Tempo: 115 BPM

Beat Strength: 65 %

Spectral Brighness: 22 %

Size of the collection: 2570 files

Number of matching clips: 4

Currently playing: chicane_saltwater.aif

Figure 4.11: Sound sliders and palettes.

circular fashion to the next sound file by pressing a button. One way to view this is that for each particular setting of slider values there is a corresponding list of sound files that have corresponding attributes. When the sliders are moved the sound is crossfaded to a new list that corresponds to the new slider settings. The extraction of the continuous attributes and their sorting is performed automatically.

In current audio software typically sliders are first adjusted, and then by pressing a submit button, files that correspond to these parameters are retrieved. Unlike this traditional use of sliders for setting parameters, sound sliders provide continuous aural feedback that corresponds to the actions of the user (direct sonification). So for example when the user sets the tempo to 150 beats-per-minute (bpm) and beat strength to its highest value there is immediate feedback about what the values represent by hearing a corresponding fast song with strong rhythm. Another important aspect about *sound sliders* is that they are not independant and the aural feedback is influenced by the settings of all of them. Figure 4.11 shows a screenshot of sound sliders used in our system.

4.7.2 Sound palettes

Sound palettes are similar to sound sliders but apply to browsing discrete attributes. A palette contains a fixed set of visual objects (text, images, shapes) that are arranged based on discrete attributes. At any time only one of the visual objects can be selected by clicking on it with the mouse. For example objects might be arranged in a table by genre and year of release. Continuous aural feedback similar to the sound sliders is supported. The bottom left corner of Figure 4.11 shows a content palette corresponding to musical genres. Sound palettes and sliders can be combined in arbitrary ways.

4.7.3 Loops

In the previously described query interfaces it is necessary to playback sound files continuously in a looping fashion. Several methods for looping are supported in the system. The simplest method is to loop the whole file with crossfading at the loop point. The main problem problem with this method is that the file might be too long for browsing purposes. For files with a regular rhythm, automatic beat detection tools are used to extract loops typically corresponding to an integer number of beats. Another approach that can be applied to arbitrary files is to loop based on spectral similarity. In this approach the file is broken to short windows and for each window a numerical representation of the main spectrum characteristics is calculated. The file is searched for windows that have similar spectral characteristics and these windows can be used to achieve smooth looping points. Finally automatic thumbnail methods that utilize more high level information can be used to extract representative short thumbnails.

4.7.4 Music Mosaics

Loops and thumbnail techniques can be used to create short representations of audio signals and collections. Another possibility for the representation of audio collections is the creation of *Music Mosaics* that are pieces created by concatenating short segments of other pieces [111, 148]. For example in order to represent a collection of songs by the Beatles, a music mosaic that would sound like Hey Jude could be created by concatenating small pieces from other Beatles songs. Time-stretching based on beat detection and overlap-add techniques can also be used in *Music Mosaics*.

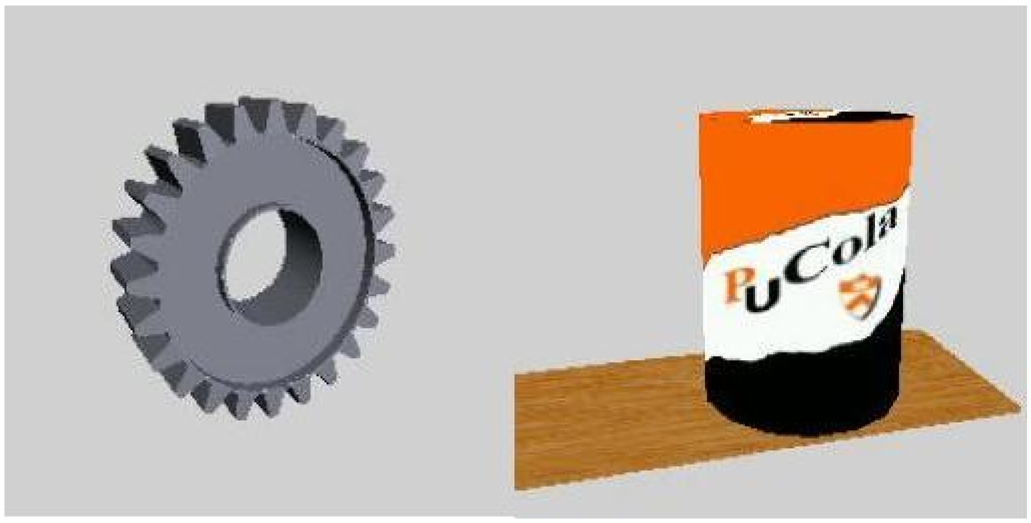


Figure 4.12: 3D sound effects generators.

4.7.5 3D sound effects generators

In addition to digital music distribution, another area where large audio collection are utilized is libraries of digital sound effects. Most of the times we hear a door opening or a telephone ringing in a movie the sounds are not actually recorded during the filming of the movie but are taken from libraries of prerecorded sound effects.

Searching libraries of digital sound effects poses new challenges to audio information retrieval. Of special interest are sound effects where the sound depends on the interaction of a human with a physical object. Some examples are: the sound of walking on gravel or the rolling of a can on a wooden table. In contrast to non-interactive sound effects such as a door bell the production of those sounds is closely tied to a mechanical motion.

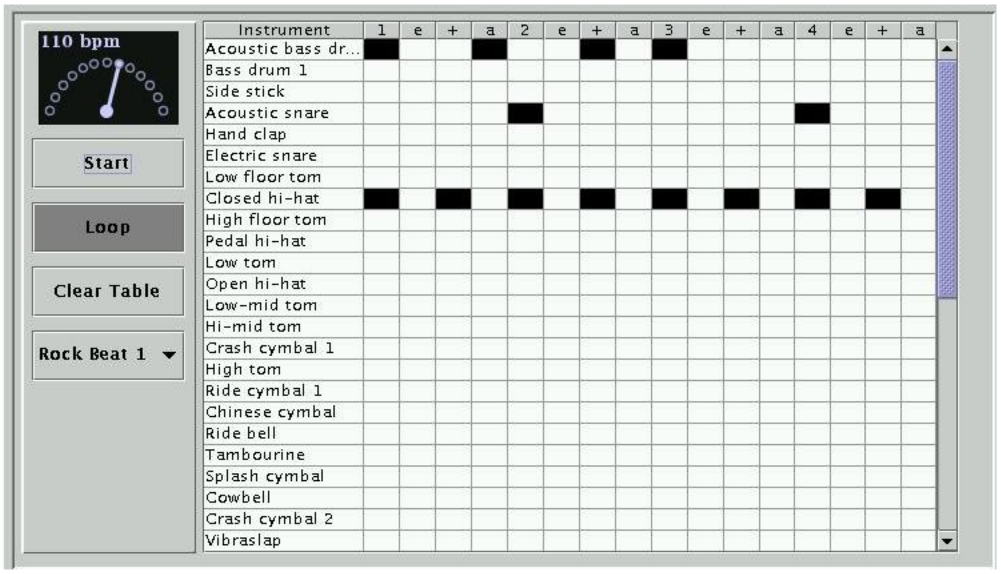
In order to go beyond the QBE paradigm for content retrieval of sound effects, query generators that somehow model the human production of these sounds are desired. Towards this goal we have deloped a number of interfaces with the common theme of providing a user-controlled animated object connected directly to the parameters of a synthesis algorithm for a particular class of sound effects. These 3D interfaces not only look similar to their real world counterparts but also sound similar. This work is part of the Physically Oriented Library of Interactive Sound Effects (PhOLISE) project [19]. This project uses physical and physically motivated analysis and synthesis algorithms such as modal synthesis, banded waveguides [33], and stochastic particle models [20] to provide interactive parametric models of real-world sound effects.

Figure 4.12 shows two such 3D sound effects query generators. The PuCola Can shown on the right is a 3D model of a soda can that can be slid across various surface textures. The sliding speed and texture material are controlled by the user and result in appropriate changes to the sound in real-time. The Gear on the left is a real-time parametric synthesis of the sound of a turning wrench. Other developed generators are: a model of falling particles on a surface where parameters such as density, speed, and material are controlled, and a model of walking sounds where parameters such as gait, heel and toe material, weight and surface material are controlled.

4.8 MIDI-based Query Interfaces

In constrast to audio based QUIs, MIDI-based QUIs do not utilize directly the audio collection but rather synthetically generate query audio signals from a symbolic representation (MIDI) that is created based on the user actions. The parameters used for the generation of this query and/or the automatically analyzed generated audio are used to retrieve similar audio pieces from a large collection. It is important to note that although MIDI is used for the generation of the queries, these interfaces are used for searching audio collections.

These concepts will be clarified in the subsequent sections with concrete examples of MIDI-based QUIs.

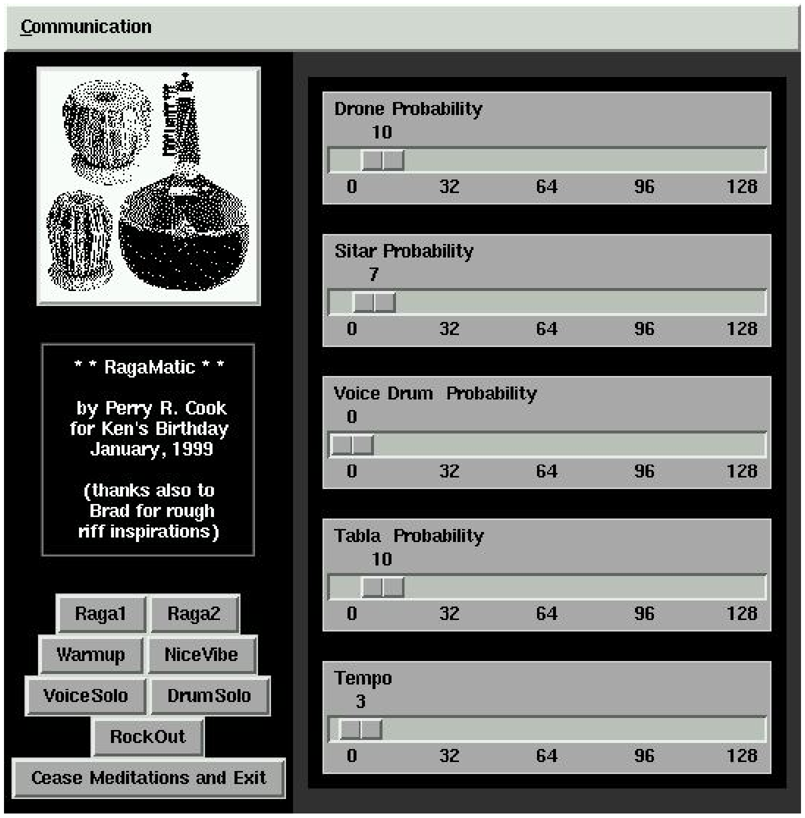


Instrument	1	e	+	a	2	e	+	a	3	e	+	a	4	e	+	a
Acoustic bass dr...	■			■			■		■							
Bass drum 1																
Side stick																
Acoustic snare					■								■			
Hand clap																
Electric snare																
Low floor tom																
Closed hi-hat	■		■		■		■		■		■		■		■	
High floor tom																
Pedal hi-hat																
Low tom																
Open hi-hat																
Low-mid tom																
Hi-mid tom																
Crash cymbal 1																
High tom																
Ride cymbal 1																
Chinese cymbal																
Ride bell																
Tambourine																
Splash cymbal																
Cowbell																
Crash cymbal 2																
Vibraslap																

Figure 4.13: Groove box

4.8.1 The groove box

The groove box is similar to standard software drum machine emulators where beat patterns beat patterns are created using a rectangular grid of note values and drum sounds. Figure 4.13 shows a screenshot of a groove machine where the user can create a beat pattern or select it from a set of predefined patterns and speed it up and down. Soundfiles are retrieved based on the interface settings as well as by audio analysis (Beat Histograms [133]) of the generated beat pattern. Another possibility is to use beat analysis methods based on extracting specific sound events (such as bass drum and cymbal hits) and match each drum track separately. The possibility of automatically alligning the generated beat pattern with the retrieved audio track and playing both as the user edits the beat pattern is under development. The code for the groove box is based on demonstration code for the JavaSound API.



* * RagaMatic * *

by Perry R. Cook
for Ken's Birthday
January, 1999

(thanks also to
Brad for rough
riff inspirations)

Drone Probability 10

Sitar Probability 7

Voice Drum Probability 0

Tabla Probability 10

Tempo 3

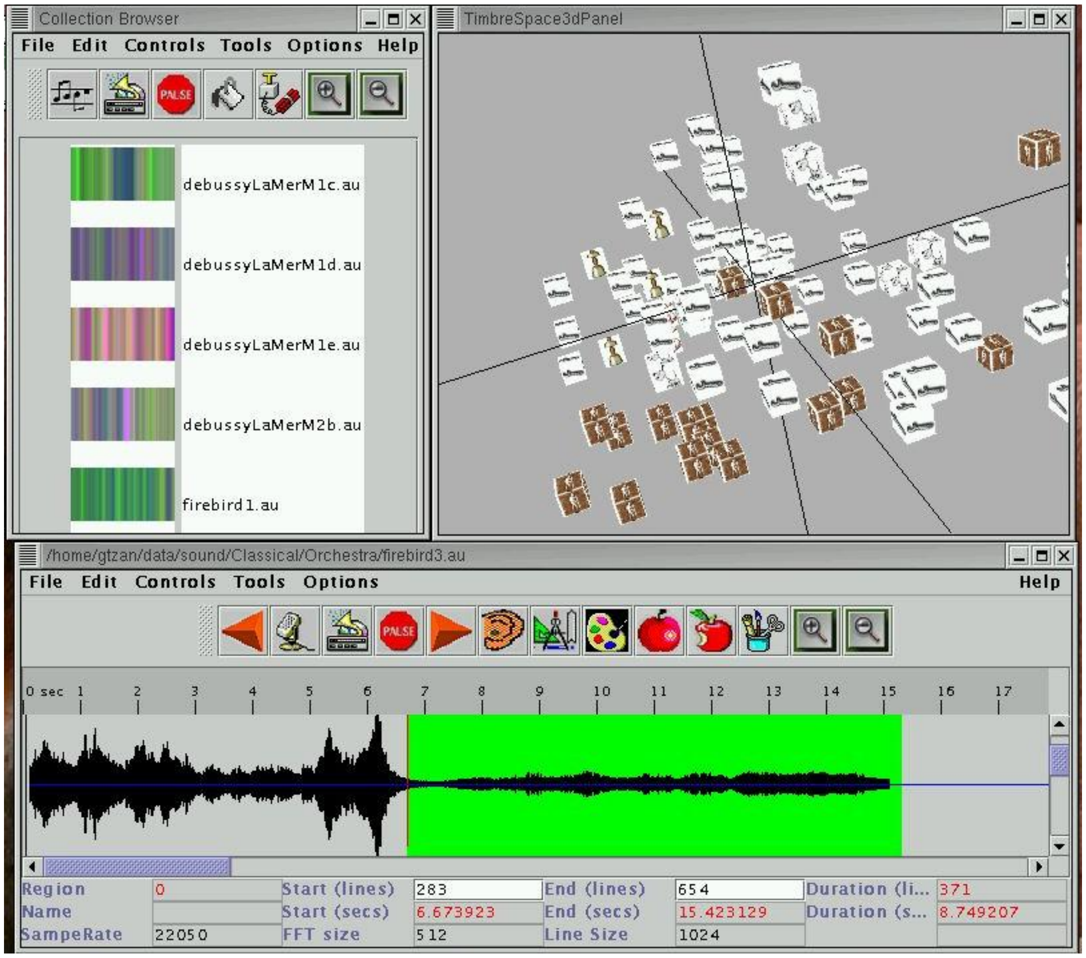
Figure 4.14: Indian music style machine

4.8.2 The tonal knob

The tonal knob shows a circle of fifths which is a ordering of the musical pitches so that harmonically related pitches are successively spaced in a circle. The user can select the tonal center and hear a chord progression at a specified musical style that establishes that tonic center. Pieces with the same tonal center are subsequently retrieved using Pitch Histograms ([133]).

4.8.3 Style machines

Style machines are more close to standard automatic music generation interfaces. For each particular style we are interested in modelling for retrieval a set of sliders and buttons corresponding to various attributes of the style are used to generate in real-time an audio signal. This signal is subsequently used as a query for retrieval purposes. Style attributes include tempo, density, key center, and instrumentation. Figure 4.14 shows a screenshot of *Ragamatic*, a style machine for Indian Music.


Collection Browser
File Edit Controls Tools Options Help
PAUSE
debussyLaMerM1c.au
debussyLaMerM1d.au
debussyLaMerM1e.au
debussyLaMerM2b.au
firebird1.au
TimbreSpace3dPanel
/home/gtzan/data/sound/Classical/Orchestra/firebird3.au
File Edit Controls Tools Options Help
PAUSE
0 sec 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17

Region	0	Start (lines)	283	End (lines)	654	Duration (li...	371
Name		Start (secs)	6.673923	End (secs)	15.423129	Duration (s...	8.749207
SampeRate	22050	FFT size	512	Line Size	1024		

Figure 4.15: Integration of interfaces

4.9 Integration

Although the described graphical user interface components were presented separately, they are all integrated following a Model-View-Controller user interface design framework [64]. The Model part comprises of the actual underlying data and the operations that can be performed to manipulate it. The View part describes specific ways the data model can be displayed and the Controller part describes how user input can be used to control the other two parts. By sharing the Model part the graphical user interface components can affect the same underlying data. That way for example *sound sliders*, *sound palettes,* and *style machines* can all be used together to browse the same audio collection visualized as a *Timbrespace* and *Timbregram* table and edited using the *Enhanced Audio Editor*. Figure 4.15 shows a snapshot of one of the possible combinations of described interfaces.

Of course in a complete music information retrieval system traditional graphical user interfaces for searching and retrieval such as keyword or metadata searching would also be used in addition to the described interfaces. These techniques and ideas are well-known and therefore are not described in this Chapter but are part of the developed system.

Although the description of the interfaces in this Chapter has emphasized music retrieval it is clear that such a system would also be useful for music creation. For example sound designers, composers (especially utilizing ideas from Music Concrete and Granular Synthesis), and DJs would all benefit from novel ways of interacting with large collections of audio signals. It is also possible that many of the proposed ideas and interfaces can be also used for browsing and interacting other types of temporal signals such as video and speech signals.

4.10 The Princeton Display Wall

Working with many audio files using the described interfaces on a desktop monitor is difficult because of the large number of windows and screen space required. In addition the desktop monitor is not appropriate for interactive collaborations among multiple simultaneous users. To overcome this difficulty, *Marsyas* has been used as an application for the Princeton Scalable Display Wall (PDW) project. The Princeton Scalable Display Wall project explores building and using a large-format display with multiple commodity

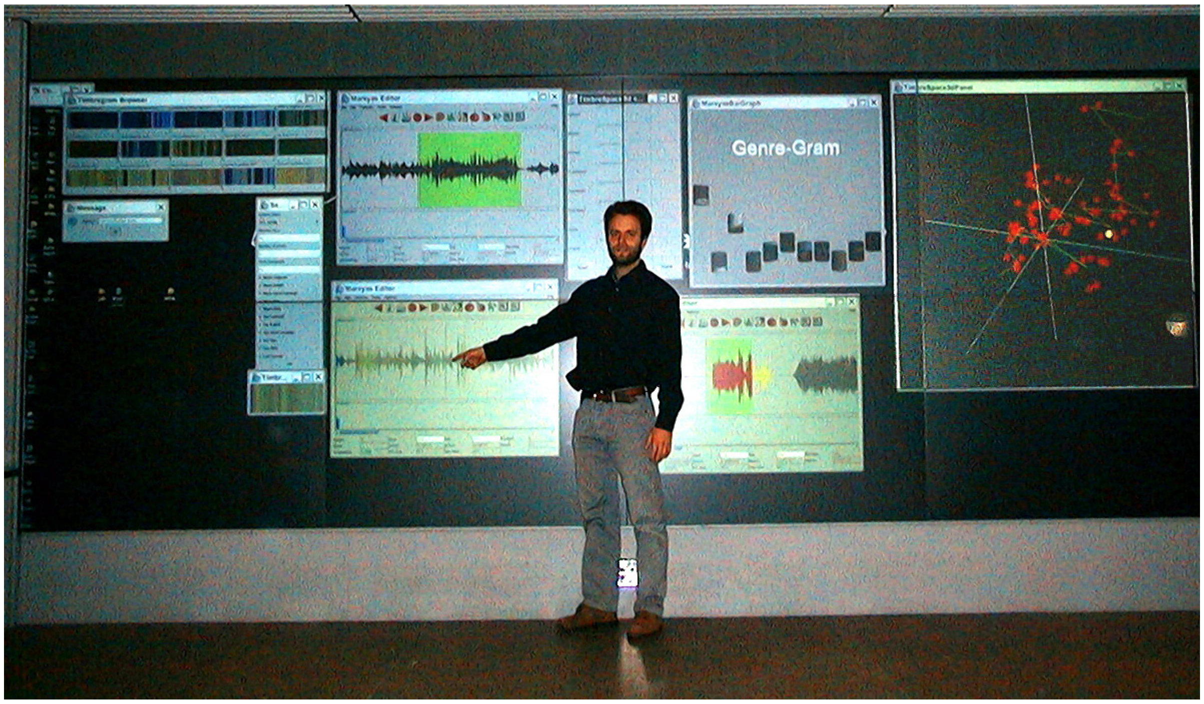


Figure 4.16: Marsyas on the Display Wall

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 [22]. A more detailed description of the project can be found [69]. 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. More information about the use of Marsyas on the PDW can found in [131] and Figure 4.9 shows a variety of the developed interfaces on the PDW.

4.11 Summary

In this Chapter, a variety of content and context aware user interfaces for browsing and interacting with audio signals and collections were presented. They are based on ideas from Visualization and Human Computer Interaction and utilize automatically extracted audio content information. Timbrespaces and Timbregrams are visualizations of collections and signals that map audio content to properties of visual objects. They can automatically adapt to audio data using automatic feature extraction and Principal Component Analysis (PCA). A series of interfaces for specifying audio queries that go beyond the traditional Query-by-Example paradigm were also presented.

Chapter 5

Evaluation

Knowledge must come through action; you can have no test which is not fanciful, save by trial.

Sophocles, Trachiniae

The first principle is that you must not fool yourself - and you are the easiest person to fool.

Richard Feynman, The Pleasure of Finding Things Out

5.1 Introduction

In the previous chapters a variety of ways for representing and analyzing audio signals and collections was presented. Evaluation of Computer Audition research as well as any type of research that attempts to model how human process and understand sensory input is difficult as there is no single correct answer. Unlike other areas in Computer Science (for example sorting algorithms) where there is a well-defined answer to the problem being solved, answers in Computer Audition are usually fuzzy and subjective in nature. That observation does not mean that evaluation of Computer Audition is impossible, however it requires carefully designed experiments that in many cases involve user studies. Another way to increase confidence for the evaluation results is to combine results from different experiments. For example, MPEG-based features will be compared to other proposed features both in the context of automatic classification and segmentation.

Another problem facing the evaluation of Computer Audition algorithms and systems it the relatively recent appearance of the field and the resulting lack of standarized evaluation data collections and results. In some cases of new types of analysis such as segmentation and thumbnailing it is not even clear how algorithms can be evaluated and evaluation experiments have to be designed from scratch.

5.2 Related Work

In a sense all the work described in this Section is a contribution as it consists of evaluation results about the algorithms and systems that have been described. These results are calculated automatically and by conducting user studies. As there are no standarized datasets in Computer Audition, direct comparisons with existing results is difficult in most cases. When it is possible to do so, direct and indirect comparisons will be provided. Therefore, the only way to evaluate and compare the performance of classifiers is by conducting empirical experiments.

5.2.1 Contributions

The basic contributions described in this Chapter are the following: 1) design of user experiments for evaluating segmentation into arbitrary sound "textures" 2) based on the conducted user experiments humans are consistent when segmenting audio, their performance can be approximated automatically and providing an editable automatic segmentation does not bias the resulting segmentation 3) MPEG-based features have comparable performance with other proposed feature sets for music/speech classification and segmentation 4) DWT-based features have comparable performance with other proposed feature sets for audio classification 5) Effective similarity retrieval is possible using timbral and beat information features 6) Beat Histograms can be used for tempo estimation and provide a good representation of rhythmic content.

The exact experiments, numbers, and details that support the previous statements will be described in the rest of this Chapter.

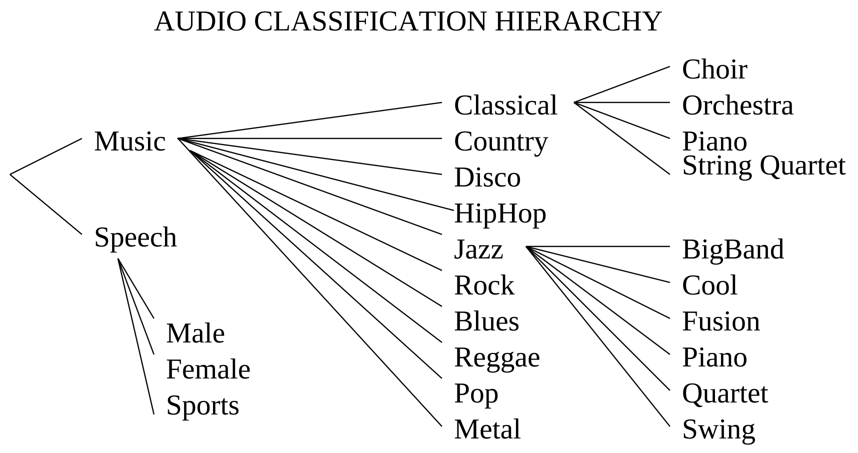


Figure 5.1: Musical Genres Hiearachy

5.3 Collections

In order to train and evaluate statistical pattern recognition classifiers it is important to have large representative datasets. In this Section the collections used to conduct the experiments described in this Chapter are described.

Figure 5.1 shows the hierachy of musical genres used for evaluation augmented by a few (3) speech related categories. In addition, a music/speech classifier similar to [110] has been implemented. For each of the 20 musical genres and 3 speech genres, 100 representative excerpts were used for training. Each excerpt was 30 seconds long resulting in (23 * 100 * 30 seconds = 19 hours) of training audio data. To ensure variety of different recording qualities the excerpts were taken from radio, compact disks, and mp3 compressed audio files. The files were stored as 22050 Hz, 16-bit, mono audio files. An effort was made to ensure that the training sets are representative of the corresponding musical genres. The Genres dataset has the following classes: Classical, Country, Disco, HipHop, Jazz, Rock, Blues, Reggae, Pop, Metal. The Classical dataset has the following classes: Choir, Orchestra, Piano, String Quartet. The Jazz dataset has the following classes: BigBand, Cool, Fusion, Piano, Quartet, Swing.

For the similarity retrieval experiment, a collection consisting of 1000 distinct 30-second rock song excerpts was used. The files were stored as 22050 Hz, 16-bit, mono audio files. In addition collections of sound effects and isolated musical instrument tones have also been used to design and develop the proposed algorithms.

5.4 Classification evaluation

Although a labeled training set is used in training automatic classifiers, we are really interested in how the classifier performs when presented with data it has not encountered before or its generalization performance. It can be shown that if the goal is to obtain good generalization performance, there are no context-independent or usage-independent reasons to favor one learning or classification method over another (*No Free Lunch Theorem*) see Chapter 9 of [30]. If one algorithm seems to outperform another in a particular situation, it is a consequence of its fit to the particular problem, not the general superiority of the algorithm. Therefore in order to evaluate and compare different classifiers it is necessary to conduct empirical studies and experiments.

5.4.1 Cross-Validation

When evaluating an automatic classification method typically there is a set of labeled data that in addition to training has to be used for testing. In simple validation the set of of labeled samples is randomly split into two sets: one is is used as the traditional training set for adjusting model parameters of the classifier. The other set, called the validation set, is used to estimate the generalization error which can be used to compare different classifiers. Obviously this error will depend on the particular split of the data to training and testing and there is the danger of overfitting the training data.

In order to avoid this problem, a simple generalization of the above method is *m-fold cross validation*. Here the training set is randomly divided into m disjoint sets of equal size n/m, where n is the total number of labeled patterns. The classifier is trained i times, each time with a different set held out as a validation set. The estimated performance is the mean of these i errors. In the limit where $m = n$, the method is called the *leave-one-out* approach.

A specific detail that is important when evaluating classifiers of audio files represented as trajectories of feature vectors is to never split feature vectors from the same file in training and testing. This is important since there is a good deal of frame-to-frame correlation in vectors of the same file so splitting the file would give an incorrect estimate of classifier performance for truly novel data. The results presented in this chapter do not split same-file feature vectors and are based on 10-fold cross-validation with 100 iterations.

5.5 Segmentation experiments

Unlike classification, where there are well-established evaluation techniques, segmentation provides a challenge. Methods that rely on previous segmentation can use the classification annotations as the ground truth however segmentation of arbitrary textures is more tricky

	Human Agreement		Automatic				
			Fixed Budget		Best Effort		
	AG	**%**	**FB**	**%**	**BE**	**MX**	**%**
Classic1	8/14	57	7/8	87	7/8	8	87
Classic2	7/11	63	5/7	71	6/7	14	85
Classic3	5/13	38	2/5	40	3/5	16	60
Jazz1	3/14	21	2/3	66	3/3	16	100
Jazz2	5/8	62	3/5	60	5/5	10	100
JazzRock	6/9	66	0/6	0	5/6	12	83
Pop1	5/6	83	4/5	80	5/5	10	100
Pop2	5/10	50	4/5	80	4/5	5	80
Radio1	4/10	40	3/4	75	4/4	10	100
Radio2	8/11	72	5/8	62	6/8	11	75
Total	**56/106**	**55**	**35/56**	**62**	**48/56**	**11**	**87**

Table 5.1: Free segmentation

and required the design of new user experiments. The main questions the designed user experiments attempt to answer are:

- Are humans consistent when segmenting audio signals ? In other words, is there a common agreement between different subjects about what constitutes a good segmentation.
- Can their performance be approximated automatically and how accurate is this approximation ?
- Does providing an editable automatic segmentation and allowing the users to modify it, bias their segmentation results ? The answer to this question is important because if it is yes then automatic segmentation can not be directly used as a preprocessing step.

Two studies [125, 128] were conducted in order to address these questions. The purpose of the first experiment was to explore what humans do when asked to segment audio and to compare those results with the automatic segmentation method.

	Human Agreement		Automatic				
			Fixed Budget		Best Effort		
	AG	**%**	**FB**	**%**	**BE**	**MX**	**%**
Classic1	4/6	66	0/4	0	4/4	10	100
Classic2	4/7	57	3/4	75	3/4	4	75
Classic3	4/7	57	2/4	50	2/4	4	50
Jazz1	3/11	27	2/3	66	3/3	16	100
Jazz2	5/6	83	3/5	60	5/5	10	100
JazzRock	5/7	71	1/5	20	4/5	12	80
Pop1	4/7	57	2/4	50	4/4	10	100
Pop2	5/7	71	4/5	80	4/5	5	80
Radio1	4/6	66	3/4	75	4/4	10	100
Radio2	5/7	71	2/5	40	4/5	10	80
Total	**43/71**	**62**	**22/43**	**51**	**37/43**	**9**	**86**

Table 5.2: 4 ± 1 segmentation

	Human Agreement		Automatic				
			Fixed Budget		Best Effort		
	AG	**%**	**FB**	**%**	**BE**	**MX**	**%**
Classic1	8/14	57	7/8	87	7/8	8	87
Classic2	7/10	70	5/7	71	6/7	14	85
Classic3	9/11	81	6/9	66	6/9	9	66
Jazz1	4/15	26	3/4	75	3/4	4	75
Jazz2	5/11	45	3/5	60	5/5	10	100
JazzRock	7/9	77	3/7	42	5/7	12	71
Pop1	6/12	50	5/6	83	6/6	10	100
Pop2	8/13	61	4/8	50	5/8	16	62
Radio1	7/10	70	3/7	42	4/7	10	57
Radio2	9/11	81	5/9	55	6/9	11	66
Total	**70/116**	**61**	**44/77**	**63**	**53/70**	**10**	**77**

Table 5.3: 8 ± 2 segmentation

The data used for both studies consists of 10 sound files about 1 minute long. A variety of styles and textures are represented. In the first study 10 subjects were asked to segment each sound file using standard audio editing tools in 3 ways. The first way, which we call free, is breaking up the file into any number of segments. The second and third way constrain the users to a specific budget of total segments 4 ± -1 and 8 ± 2.

The results of the first study are shown in Tables 5.1,5.2 and 5.3. The segments that more than 5 of the 10 subjects agreed upon were used for comparison. The AG column shows the number of these salient segments compared to the total number of all segments marked by any subject. It is a measure of consistency between the subjects. For comparing the automatic method a segment boundary was considered to be the same if it was within 0.5 sec of the average human boundary. This was based on the deviation of segment boundaries between subjects. FB (fixed-budget) refers to automatic segmentation by requesting the same number of segments as the salient human segments. BE (best effort) refers to the best automatic segmentation achieved by incrementally increasing the number of regions up to a maximum of 16. MX is the number of segments necessary to achieve the best effort segmentation.

The results show that humans are consistent when segmenting audio (more than half of the segments are common for most of the subjects). In addition, they show that human segmentation can be approximated by automatic algorithms. The biggest problem seems to be the perceptual weighting of a texture change. For example, many errors involved soft speech entrances that were marked by all the subjects although they were not significant as changes in the computed feature space. The automatic segmentation results are usually perceptually justified and a superset of the human segmented regions. In the first study subjects were allowed to use the sound editor of their choice, therefore no timing information was collected.

In the second study [128] a new set of user experiments were conducted to re-confirm the results of [125] and to answer some additional questions. The main question we tried to answer was if providing an automatic segmentation as a basis for editing would bias the resulting segmentation. In addition information about the time required to segment and annotate audio was collected. The use of automatically segmented time-lines can greatly accelerate the segmentation and annotation process. Unlike the first study [125] where participants could use a sound editor of their choice, in the second study they used a segmentation-annotation tool based on the *Enhanced Sound Editor* described in chapter 4. The segmentation-annotation tool consists of a graphical user interface looking like a typical sound-editor. Using a waveform display arbitrary regions for playback can be selected and annotated time lines can be loaded and saved. Each segmented region is colored different and the user can move forward and backward through those regions. In addition to the typical sound editor functionality the system can automatically segment the audio to suggest a time line. The resulting regions can then be edited by adding/deleting boundaries until the desired segmentation is reached.

In Figure 5.2 histograms of subject agreement are shown. To calculate the histogram all the segmentation marks were collected and partitioned into bins of agreement in the following manner: all the segment marks within $\pm 0.5 sec$ were considered as corresponding to the same segment boundary. This value was calculated based on the differences between the exact location of the segment boundary between subjects and was confirmed by listening to the corresponding transitions. Since at most all the ten subjects can contribute marks within this standard deviation the maximum number of subject agreement is 10. In the figure the different lines show the histogram of the experiments in [125] (old), the results of the second study (new) and the total histogram of both studies (total). The total histogram is calculated by considering the 20 subjects and dividing by two to normalize (notice that this is different than taking the average of the two histograms since this is

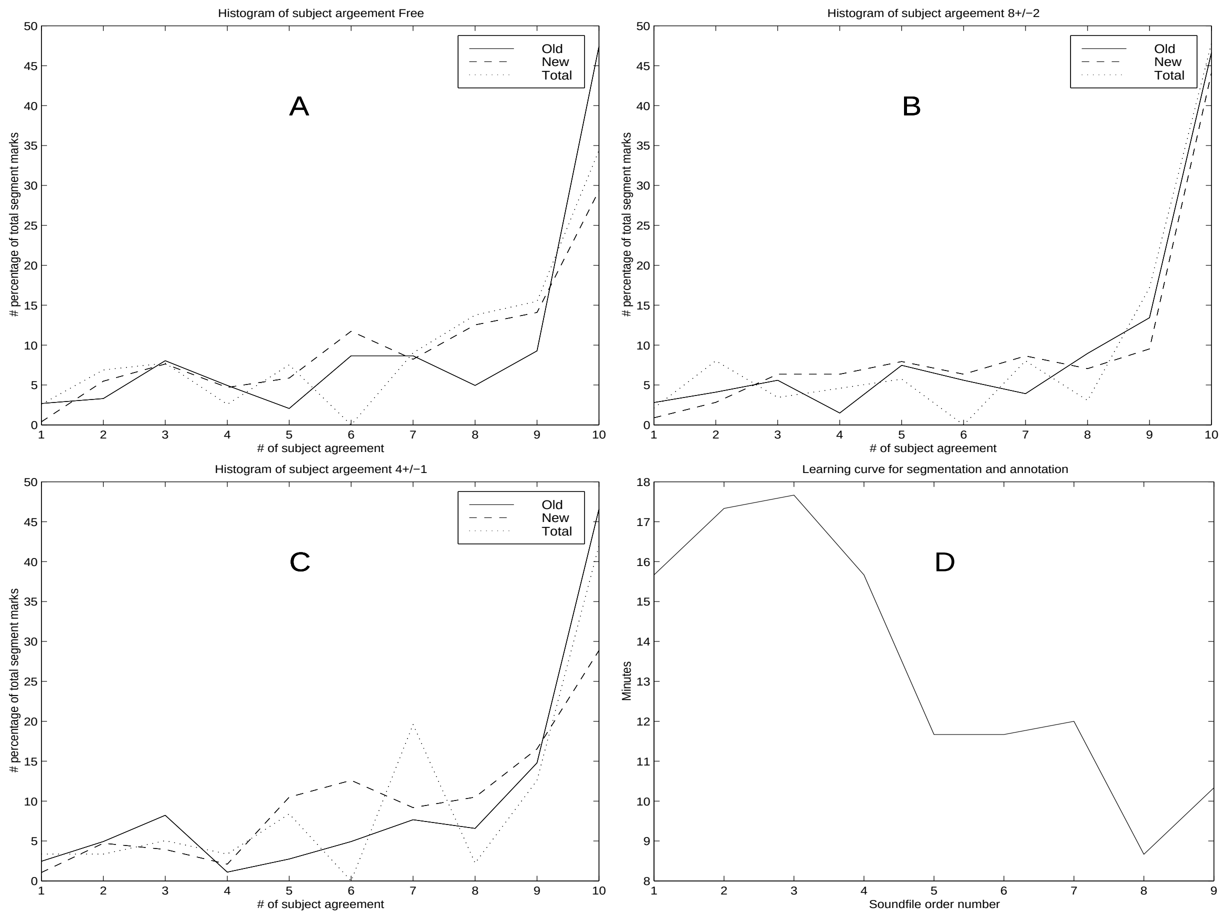


Figure 5.2: Histograms of subject agreement and learning curve for mean segment completion time

done on a boundary basis). Therefore if in the two studies the boundaries were changed indicating bias because of the automatically suggested segmentation, the shape of the total histogram would drastically change. The histograms show that a there is a large percentage of agreement between subjects and that the segmentation results are not affected by the provision of an automatically suggested segmentation time line. Finally the fact that the histograms of the old,new and total experiments have about the same shape in free, 8 ± 2 and 4 ± 1 suggests that constraining the user to a specific budget of segments does not affect significantly their agreement.

As a metric of subject agreement we define the percentage of the total segmentation marks that more than 5 of the 10 subjects agreed upon. This number can be calculated by integrating the histograms from 6 to 10. For the experiments in [125] this metric gives 79%, for this study 76% and, for the combined total 73%. In [125], 87% of the segments than half the subjects agreed upon were in the set of the best effort automatic segmentation. Moreover for this study (new) 70% of the total segment marks by all subjects were retained from the best effort automatically suggested segmentation. All these numbers are for the case of free segmentation.

The mean and standard deviation of the time it took to complete (segment and annotate) a soundfile with duration of about 1 minute was 13 ± 4 minutes. This result was calculated using only the free segmentation timing information because the other cases are much faster due to the familiarity with the soundfile and the reusability of segmentation information from the free case. The lower right sub-figure of Figure 5.2 shows the average time per soundfile in order of processing. The order of processing was random therefore the figure indicates there is a significant learning curve for the task. This happens despite the fact that an initial soundfile that was not timed was used to familiarize the users with the interface. Therefore the actual mean time is probably lower (about 10 minutes) for an experienced user.

5.6 MPEG-based features

The performance of features calculated from MPEG-compressed data is evaluated and compared with other feature sets in the context automatic music/speech classification and segmentation.

5.6.1 Music/Speech classification

The data used for evaluating the Music/Speech classification 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. The results are calculated using 10-fold cross validation as described in Section 5.4.1. The results are calculated on a frame basis without any integration. Further improvements in classification performance can be achieved by integrating the results. For integration the region detected by the segmentation algorithm can be used.

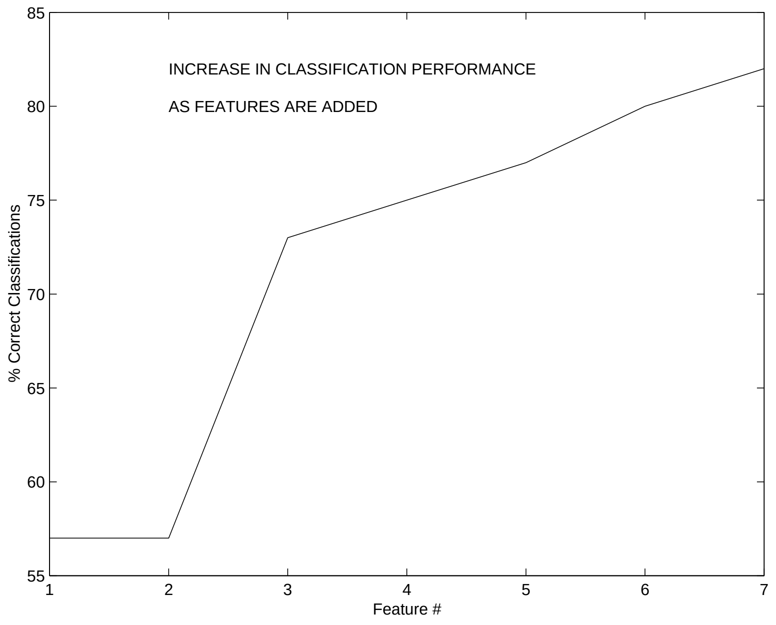


Figure 5.3: 1,2,3 and 4,5,6 are the means and variances of MPEG Centroid,Rolloff and Flux. Low Energy is 7.

Figure 5.3, shows the improvement in classification performance by the gradual addition of features. The order used is *mean Centroid, mean Rolloff, mean Flux, variance Centroid, variance Rolloff, variance Flux and Low Energy*. Table 1 compares the performance of the MPEG based classification with PCM based systems that use Spectral Shape

	GMAP	K-NN(1)	K-NN(5)
MPEG	$82.2 \pm 2.1\%$	$84.6 \pm 4.4\%$	$86.2 \pm 3.8\%$
PCM	$84.8 \pm 1.4\%$	$89.5 \pm 2.0\%$	$90.0 \pm 1.2\%$
Different Data Set			
OTHER1	$94.0 \pm 2.6\%$	$94.2 \pm 3.6\%$	$95.7 \pm 3.5\%$
Different Data Set & Classifiers			
OTHER2	77.1%	90.0%	94.2%

Table 5.4: Music/Speech percentages of frame-based classification performance

	Free	4 ± 1	8 ± 2
MPEG FB	55%, 31/56	58%, 25/43	54%, 38/70
PCM FB	62%, 35/56	51%, 22/43	62%, 44/70
MPEG BE	71%, 40/56	74%, 32/43	65%, 46/70
PCM BE	85%, 48/56	86%, 37/43	75%, 53/70

Table 5.5: Comparison of segmentation algorithms

Features calculated using the STFT. For the PCM line the same dataset and a similar set of features calculated using short time Fourier Transform were used. The OTHER1 line is based on results by [110] and uses a different data set. The OTHER2 line is based on results by [102] and uses a different data set and different classifiers. The results indicate that the proposed features can be used for classification without significant decrease in accuracy. These results are calculated using a GMAP classifier. Because the main purpose is to compare the MPEG feature set with other feature sets the particular choice of classifier is not important.

5.6.2 Segmentation evaluation

The comparison of MPEG-based features to PCM-based features for automatic segmentation is based on the experiments described in Section 5.5. In table 5.6.2 the performance of the MPEG-based segmentation algorithm is compared with a similar algorithm based on short time Fourier Transform analysis. The table shows the number of regions that were marked by humans that were captured by the system. FB (fixed-budget) refers to automatic segmentation by requesting the same number of segments as the salient human segments. BE (best effort) refers to the best automatic segmentation achieved by incrementally increasing the number of regions up to a maximum of 16. Although the performance is lower than the PCM based algorithm still a large number of segmentation boundaries are captured by the algorithm. Most of the cases where the algorithm missed boundaries compared to the PCM based algorithm were in soundfiles containing rock or jazz music. The reason is that the PCM based algorithm uses time domain zerocrossings that capture the transition from noise-like signals to more harmonic signals.

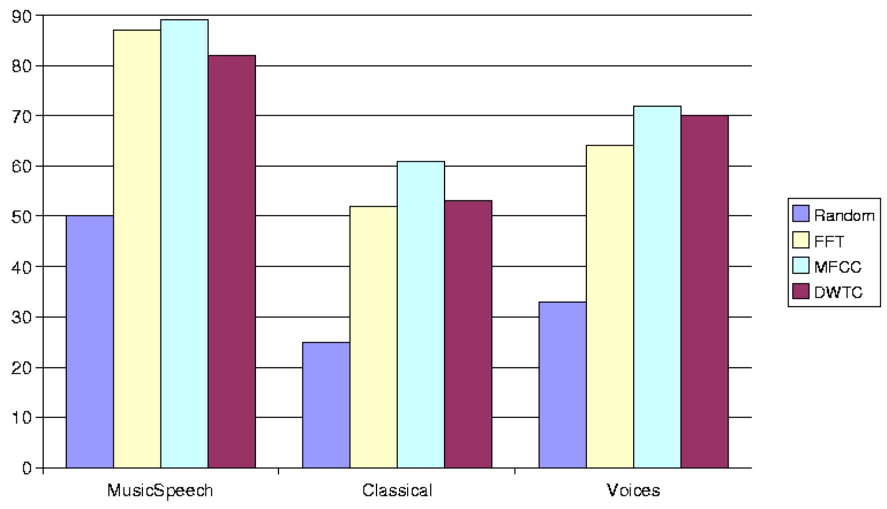


Figure 5.4: DWT-based feature evaluation based on audio classification

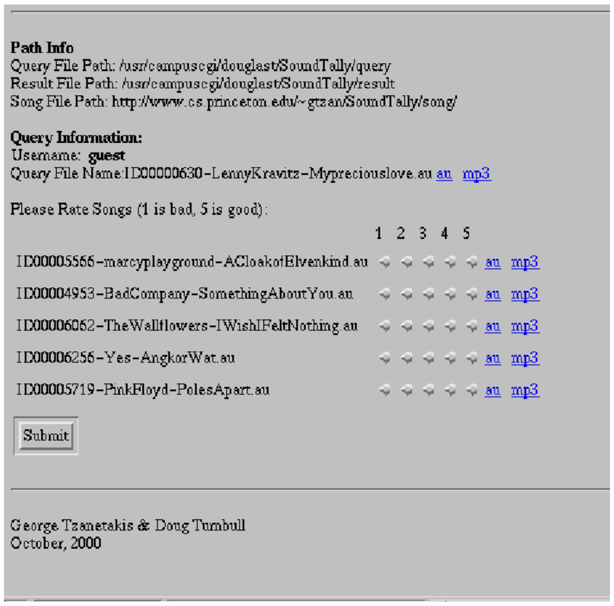
5.7 DWT-based features

The features based on the DWT transform were evaluated in the context of audio classification and more specifically using the following classification schemes: Music/Speech, Classical Genres, and Voices. Figure 5.4 and Table 5.7 compare the classification results

	MusicSpeech	Classical	Voices
Random	50	25	33
FFT	87	52	64
MFCC	89	61	72
DWTC	82	53	70

Table 5.6: DWT-based feature evaluation based on audio classification

between FFT-based, MFCC, and DWT-based features. These results are calculated using a GMAP classifier. Because the main purpose is to compare the performance of the DWT-based feature set with other feature sets, the particular choice of classifier is not important.



Path Info
Query File Path: /usr/campuscgi/douglast/SoundTally/query
Result File Path: /usr/campuscgi/douglast/SoundTally/result
Song File Path: http://www.cs.princeton.edu/~gtzan/SoundTally/song/

Query Information:
Username: **guest**
Query File Name:ID00000630-LennyKravitz-Mypreciouslove.au au mp3

Please Rate Songs (1 is bad, 5 is good):

	1	2	3	4	5		
ID00005566-marcyplayground-ACloakofElvenkind.au	○	○	○	○	○	au	mp3
ID00004953-BadCompany-SomethingAboutYou.au	○	○	○	○	○	au	mp3
ID00006062-TheWallflowers-IWishIFeltNothing.au	○	○	○	○	○	au	mp3
ID00006256-Yes-AngkorWat.au	○	○	○	○	○	au	mp3
ID00005719-PinkFloyd-PolesApart.au	○	○	○	○	○	au	mp3

George Tzanetakis & Doug Turnbull
October, 2000

Figure 5.5: Web-based Retrieval User Evaluation

5.8 Similarity retrieval evaluation

In order to perform similarity retrieval user studies, a Web evaluation infastructure was developed. Figure 5.5 shows a sample page for evaluating the results of a query. A user survey of similarity retrieval was conducted using the *Rock* dataset (1000 30-second rock songs). The large size of the dataset made the calculation of recall difficult so only

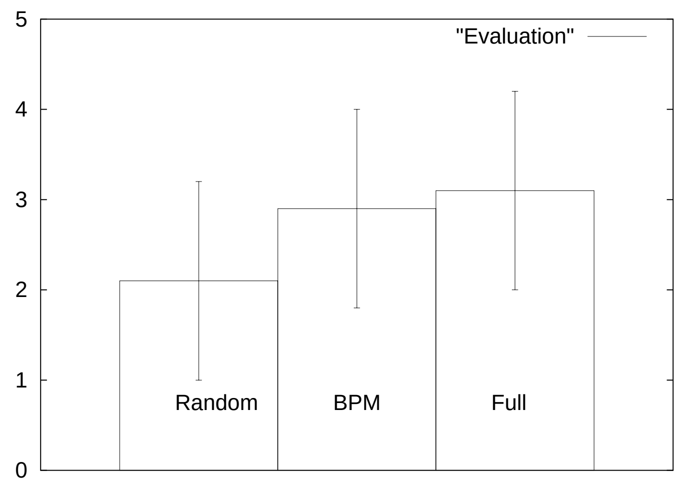


Figure 5.6: Results of User Evaluation

precision was examined. Because the 30-second snippets used for the evaluation did not have many texture changes, the single vector approach was used. The evaluation was done by 7 subjects who gave a relevance judgement from 1 (worse) to 5 (best). There were twelve queries, five matches returned and three algorithms (random, only beat-detection, beat and texture) giving a total of 7 * 5 * 12 * 3 = 1260 data points. The beat detection was done using the algorithm described in [108]. In this algorithm, a filterbank is coupled with a network of comb filters that track the signal periodicities to provide an estimate of the main beat and its strength. The full retrieval was done using both beat detection and spectral features.

Figure 5.6 shows the mean and standard deviation of the three retrieval methods. The standard deviation is due to the different nature of the queries and subject differences and is about the same for all algorithms. Although it is clear that the system performs better than random and that the full approach is slightly better than using only beat detection more work needs to be done to improve the scores. Significantly better results are achieved using the full musical content feature sets that includes Beat and Pitch Histogram features.

Unfortunately, this study was conducted before the design of these new features. A similar evaluation study using the newly proposed features is planned for the future.

5.9 Automatic Musical Genre Classification

Because this is one of the first published works that deal seriously with the problem of automatic musical genre classification a variety of experimental results will be presented.

5.9.1 Results

Table 5.7 shows the classification accuracy percentage results of different classifiers and musical genre datasets. With the exception of the RT GS row, these results have been computed using a single-vector to represent the whole audio file. The vector consists of the timbral texture features (9 (FFT) + 10 (MFCC) = 19-dimensions), the rhythmic content features (6 dimensions), and the pitch content features (5 dimensions) resulting in a 30-dimensional feature vector. In order to compute a single timbral-texture vector for the whole file the mean feature vector over the whole file is used.

Table 5.7: Classification accuracy mean and standard deviation

	Genres(10)	Classical(4)	Jazz(6)
Random	10	25	16
RT GS	44 ± 2	61 ± 3	53 ± 4
GS	59 ± 4	77 ± 6	61 ± 8
GMM(2)	60 ± 4	81 ± 5	66 ± 7
GMM(3)	61 ± 4	88 ± 4	68 ± 7
GMM(4)	61 ± 4	88 ± 5	62 ± 6
GMM(5)	61 ± 4	88 ± 5	59 ± 6
KNN(1)	59 ± 4	77 ± 7	57 ± 6
KNN(3)	60 ± 4	78 ± 6	58 ± 7
KNN(5)	56 ± 3	70 ± 6	56 ± 6

The row RT GS shows classification accuracy percentage results for real-time classification per frame using only the timbral texture feature set (19-dimensions). In this case each file is represented by a time series of feature vectors, one for each *analysis window*. Frames from the same audio file are never split between training and testing data in order to avoid false higher accuracy due to the similarity of feature vectors from the same file. A comparison of random classification, real-time features, and whole-file features is shown in Figure 5.7. The data for creating this bar graph corresponds to the Random, RT GS, and GMM(3) rows of Table 5.7.

The classification results are calculated using a 10-fold cross-validation evaluation where the dataset to be evaluated is randomly partitioned so that 10% is used for testing and 90% is used for training. The process is iterated with different random partitions and the results are averaged (for Table 5.7 100 iterations were performed). This ensures that the calculated accuracy will not be biased because of a particular partitioning of training and testing. If the datasets are representative of the corresponding musical genres then these results are also indicative of the classification performance with real-world unknown signals. The $\pm$ part shows the standard deviation of classification accuracy for the iterations. The row labeled *Random* corresponds to the classification accuracy of a chance guess.

The additional music/speech classification has 86% (random would be 50%) accuracy and the voices classification (male, female, sports announcing) has 74% (random 33%). Sports announcing refers to any type of speech over a very noisy background. The STFT-based feature set is used for the music/speech classification and the MFCC-based feature set is used for the speech classification.

Confusion matrices

Table 5.9 shows more detailed information about the musical genre classifier performance in the form of a confusion matrix. In a confusion matrix, the columns correspond to the

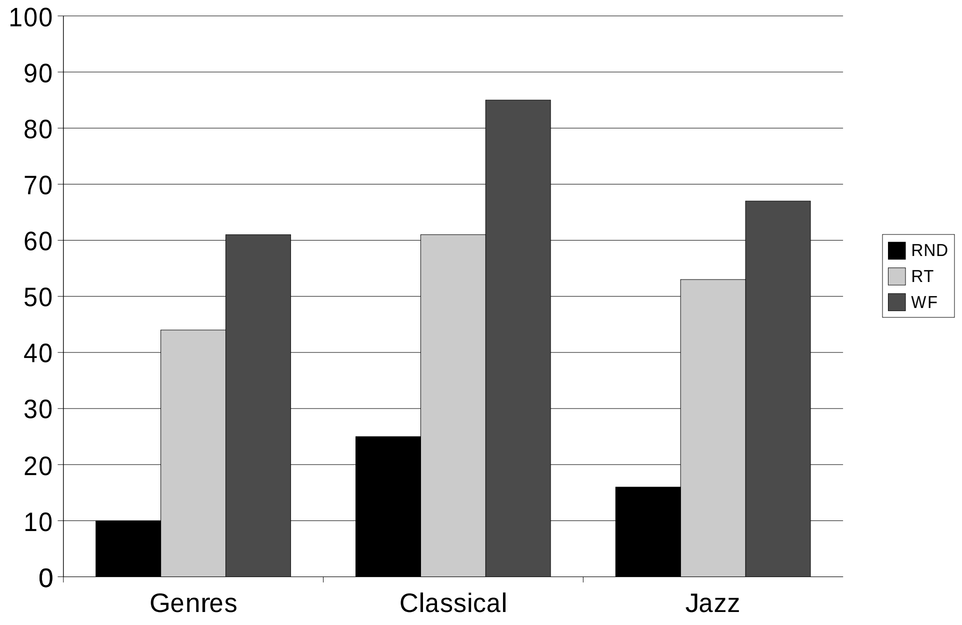


Figure 5.7: Classification accuracy percentages (RND=random,RT=real time, WF=whole file)

actual genre and the rows to the predicted genre. For example, the cell of row 5, column 1 with value 26 means that 26% of the Classical Music (column 1) was wrongly classified as Jazz music (row 2). The percentages of correct classification lie in the diagonal of the confusion matrix. The confusion matrix shows that the misclassifications of the system are similar to what a human would do. For example *Classical* music is misclassified as *Jazz* music for pieces with strong rhythm from composers like Leonard Bernstein, and George Gershwin. *Rock* music has the worst classification accuracy and is easily confused with other genres which is expected because of its broad nature.

Tables 5.10, 5.11 show the confusion matrices for the Classical and Jazz genre datasets. In the Classical genre dataset, *Orchestral* music is mostly misclassified as *String Quartet*. As can be seen from the confusion matrix (Table III 5.10), Jazz genres are mostly mis-

Table 5.8: Real Time Classification accuracy percentage results

	Genres(10)	Classical(4)	Jazz(6)
Random	10	25	16
Gaussian	54 ± 4	77 ± 7	62 ± 8
GMM(3)	57 ± 4	85 ± 6	70 ± 7

Table 5.9: Genre Confusion Matrix

	cl	co	di	hi	ja	ro	bl	re	po	me
cl	**69**	0	0	0	1	0	0	0	0	0
co	0	**53**	2	0	5	8	6	4	2	0
di	0	8	**52**	11	0	13	14	5	9	6
hi	0	3	18	**64**	1	6	3	26	7	6
ja	26	4	0	0	**75**	8	7	1	2	1
ro	5	13	4	1	9	**40**	14	1	7	33
bl	0	7	0	1	3	4	**43**	1	0	0
re	0	9	10	18	2	12	11	**59**	7	1
po	0	2	14	5	3	5	0	3	**66**	0
me	0	1	0	1	0	4	2	0	0	**53**

classified as *Fusion*. This is due to the fact that *Fusion* is a broad category that exhibits large variability of feature values. *Jazz Quartet* seems to be a particularly difficult genre to correctly classify using the proposed features (it is mostly misclassified as *Cool* and *Fusion*).

Importance of texture window size

Figure 5.8 shows how changing the size of the *texture window* affects the classification performance. It can be seen that the use of a *texture window* increases significantly the classification accuracy. The value of zero *analysis windows* corresponds to using directly the features computed from the *analysis window*. After approximately 40 *analysis windows* (1 second) subsequent increases in texture window size do not improve classification as

Table 5.10: Jazz Confusion Matrix

	BBand	Cool	Fus.	Piano	4tet	Swing
BBand	**42**	2	1	0	6	1
Cool	21	**67**	5	4	23	10
Fus.	28	16	**88**	0	38	22
Piano	1	0	0	**80**	0	0
4tet	4	5	2	0	**19**	5
Swing	4	10	4	16	14	**62**

Table 5.11: Classical Confusion Matrix

	Choir	Orch.	Piano	Str.4tet
Choir	**99**	7	7	3
Orch.	0	**58**	2	7
Piano	0	9	**86**	4
Str.4tet	1	26	5	**86**

they don't provide any additional statistical information. Based on this plot, the value of 40 analysis windows was chosen as the *texture window* size. The timbral-texture feature set (STFT and MFCC) for the whole file and a single Gaussian classifier (GS) were used for the creation of Figure 5.8.

Importance of individual feature sets

Table 5.12 shows the individual importance of the proposed feature sets for the task of automatic musical genre classification. As can be seen the non-timbral texture features (Pitch Histogram Features (PHF) and Beat Histogram Features (BHF) perform worse than the timbral-texture features (STFT, MFCC) in all cases. However in all cases the proposed feature sets perform better than random classification therefore provide some information about musical genre and therefore musical content in general. The last row of table 5.12 corresponds to the full combined feature set and the first row corresponds to random

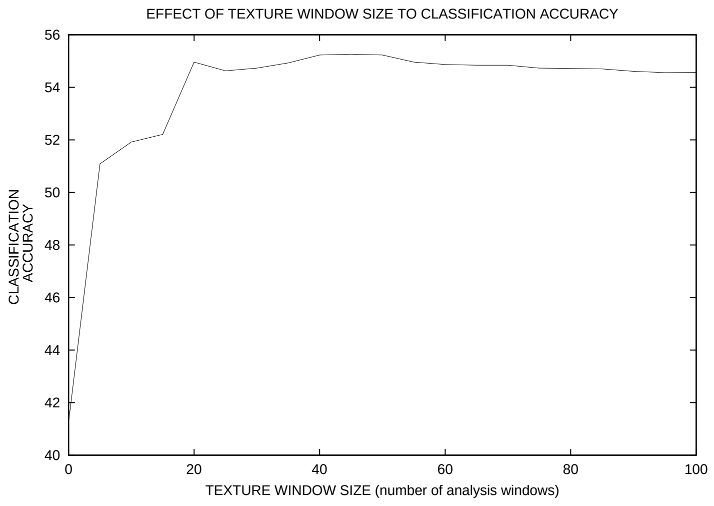


Figure 5.8: Effect of texture window size to classification accuracy

classification. The number in parentheses beside each feature set denotes the number of individual features for that particular feature set. The results of table 5.12 were calculated using a single Gaussian classifier (GS) using the whole-file approach.

Table 5.12: Individual feature set importance

	Genres	Classical	Jazz
RND	10	25	16
PHF(5)	23	40	26
BHF(6)	28	39	31
STFT(9)	45	78	58
MFCC(10)	47	61	56
FULL(30)	59	77	61

The classification accuracy of the combined feature set, in some cases, is not significantly increased compared to the individual feature set classification accuracies. This fact does not necessarily imply that the features are correlated or do not contain useful information because it can be the case that a specific file is correctly classified by two different feature sets that contain different and uncorrelated feature information. In addition although certain individual features are correlated, the addition of each specific feature improves classification accuracy. The rhythmic and pitch content feature sets seem to play a less important role in the Classical and Jazz dataset classification compared to the Genre dataset. This is an indication that it is possible that genre-specific feature sets need to be designed for more detailed subgenre classification.

Table 5.13: Best individual features

	Genres
BHF.SUM	20
PHF.FP0	23
STFT.VCTRD	29
MFCC.MMFCC1	25

Table 5.13 shows the best individual features for each feature set. These are the sum of the Beat Histogram (BHF.SUM), the period of the first peak of the Folded Pitch Histogram (PHF.FP0), the variance of the Spectral Centroid over the texture window (STFT.FPO) and the mean of the first MFCC coefficient over the texture window (MFCC.MMFCC1).

5.9.2 Human performance for genre classification

The performance of humans in classifying musical genre has been investigated in [92] [30]. Using a ten-way forced choice paradigm college students were able to accurately judge (53% correct) after listening to only 250 milliseconds samples and (70% correct) after listening to three seconds (chance would be 10%). Listening to more than 3 seconds did not improve their performance. The subjects where trained using representative samples from each genre. The ten genres used in this study were: Blues, Country, Classical, Dance, Jazz, Latin, Pop, R & B, Rap and Rock. Although direct comparison of these results with the automatic musical genre classification results is not possible due to different genres and datasets, it is clear that the automatic performance is not far away from the human performance. Moreover these results indicate the fuzzy nature of musical genre boundaries.

5.10 Other experimental results

In this Section some additional experimental results collected in the course of the previously described user studies are presented.

5.10.1 Audio Thumbnailing

An additional component of the annotation tasks of [125] was that of ”thumbnailing.” After doing the free, 8+/-2, and 4+/-1 segmenting tasks, the subjects were instructed to note the begin and end times of a two second thumbnail segment of audio that best represented each section of their free segmentation sections.

Inspection of the 545 total user thumbnail selections reveals that 62% of them were chosen to be the first two seconds of a segment, and 92% of them were a selection of two seconds within the first five seconds of the segment. This implies that a machine algorithm which can perform segmentation could also do a reasonable job of matching human performance on thumbnailing. By simply using the first five seconds of each segment as the thumbnail, and combining with the results of the best-effort machine segmentation (87%) match with human segments), a set containing 80% ”correct” thumbnails could be automatically constructed.

5.10.2 Annotation

Some work in verbal cues for sound retrieval [140] has shown that humans tend to describe isolated sounds by source type (what it is), situation (how it is made), and onomatopoeia (sounds like). Text labeling of segments of a continuous sound stream might be expected to introduce different description types, however. In the conducted experiments, a preliminary investigation of semantic annotations of sound segments was conducted. While doing the segmentation tasks, subjects were instructed to ”write a short (2-8 words) description of the section...” The annotations from the free segmentations were inspected by sorting the words by frequency of occurrence.

The average annotation length was 4 words, resulting in a total of 2200 meaningful (words like of, and, etc. were removed) words, and 620 unique words. Of these, only 100 words occur 5 or more times and these represent 64% of the total word count. Of these ”top 100” words, 37 are literal descriptions of the dominant source of sound (piano, female, strings, horns, guitar, synthesizer, etc.), and these make up almost 25% of the total words used.

Even though only 5 of the 20 subjects could be considered professional composers/musicians, then next most popular word type could be classed as music-theoretical structural descriptions (melody, verse, sequence, tune, break, head, phrase, etc.) 29 of the top 100 words were of this type, and they represent 23% of the total words used.

Another significant category of words used corresponded to basic acoustic parameters (soft, loud, slow, fast, low, high, build, crescendo, increase, etc.). Most of such parameters are easy to calculate from the signal. 12 of these words represented about 10% of the total words used.

These preliminary findings indicate that with suitable algorithms determining basic acoustical parameters (mostly possible today), the perceptually dominant sound source type (somewhat possible today), and music-theoretical structural aspects of sounds segments (much algorithmic work still to be done), machine labeling of a fair number of segments (60%) would be possible.

5.10.3 Beat studies

A number of synthetic beat patterns were constructed to evaluate the Beat Histogram calculation. Bass-drum, snare, hi-hat and cymbal sounds were used, with addition of woodblock and three tom-tom sounds for the unison examples. The simple unison beats

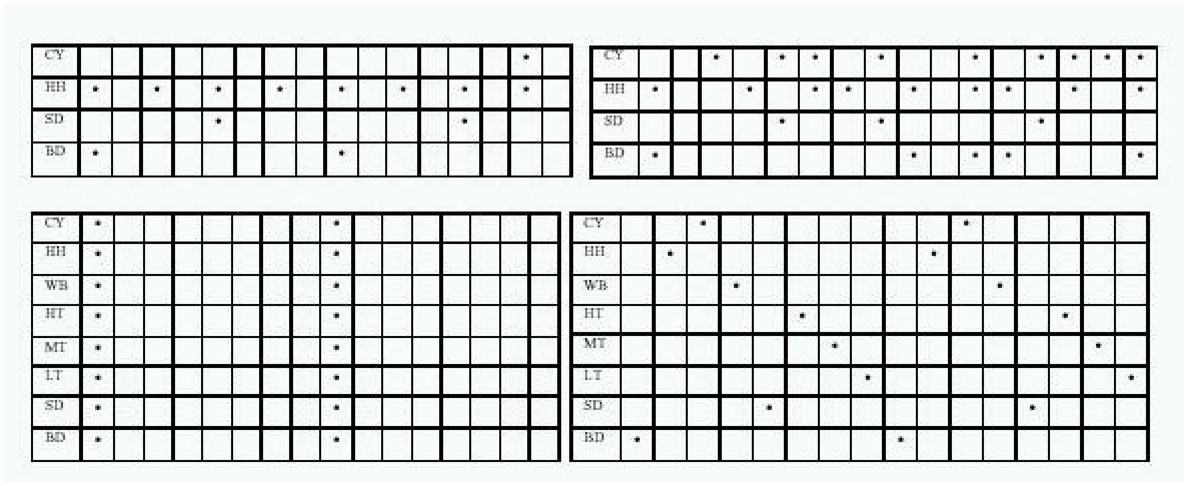


Figure 5.9: Synthetic Beat Patterns (CY = cymbal, HH = hi-hat, SD = snare drum, BD = base drum, WB = wood block, HT/MT/LT = tom-toms)

versus cascaded beat patterns are used to check for phase sensitivity in the algorithm. Simple and complex beat patterns are used to gauge the possibility to find the dominant peak timing. For this purpose, the beat patterns have been synthesized at 5 different speeds. The simple rhythm pattern has a simple rhythmic substructure with single-beat regularity, whereas the complex pattern shows a complex rhythmic substructure across and within the single instrument sections. Figure 5.9 shows the synthetic beat patterns that were used for evaluation in tablature notation. In all the cases the algorithm detected the basic beat of the pattern as a prominent peak in the Beat Histogram.

In addition to the synthetic beat patterns the algorithm was applied to real world music signals. To evaluate the algorithm's performance it was compared to the beats-per-minute (bpms) detected manually by tapping the mouse with the music. The average time difference between the taps was used as the manual beat estimate. Twenty files containing a variety of music styles were used to evaluate the algorithm (5 HipHop, 3 Rock, 6 Jazz, 1 Blues, 3 Classical, 2 World). For most of the files (13/20) the prominent beat was detected clearly (i.e the beat corresponded to highest peak of the histogram). For (5/20) files the beat was detected as a histogram peak but it was not the highest, and for (2/20) no peak corresponding to the main beat was found. In the pieces that the beat was not detected there was no dominant periodicity (either classical or free jazz). In such cases humans rely on more high-level information like grouping, melody and harmonic progression to perceive the primary beat from the interplay of multiple periodicities.

5.11 Summary

A series of experimental results regarding the evaluation of the proposed *Computer Audition* algorithms and systems were described. Using a newly designed user study for segmentation evaluation, it was shown that humans are consistent when segmenting audio and their results can be approximated automatically. In addition, the results of a user study evaluating similarity retrieval show that spectral and beat information are important for music retrieval. It was shown that automatic musical genre classification is possible with results that are not far away from the performance reported for human subjects. Finally evidence was presented that Beat Histograms provide a good representation of rhythmic content.

Chapter 6

Implementation

In the development of the understanding of complex phenomena, the most powerful tool available to the human intellect is abstraction. Abstraction arises from the recognition of similarities between certain objects, situations, or processes in the real world and the decision to concentrate on these similarities and to ignore, for the time being, their differences.

C.A.R. Hoare, Structured Programming

6.1 Introduction

In the previous chapters a variety of different algorithms and systems for manipulation, analysis and retrieval of audio signals were presented. Although for presentation purposes they were described separately, in reality they share many common building blocks and have to be integrated to create useful tools for interacting with audio signals and collections.

The main subject of this chapter is **Marsyas**, which is a free software framework for rapid prototyping of Computer Audition research that we have developed. A framework is set of cooperating classes that make up a reusable design for a specific class of software [55]. A framework is similar to a library in that it provides a set of reusable components and building blocks that can assembled into complex system. In addition, it has to be flexible and extensible allowing the addition and integration of new components with minimal effort. Finding the right abstractions and making the framework flexible and extensible is not trivial. Computer Audititition systems use similar features, analysis algorithms, and interfaces for a variety of different taks. Therefore, in the design of our system, an effort was made to abstract these 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.

The desired properties of a good framework require careful programming design. An additional complication, in the case of a framework for Computer Audition research, is the requirement for efficient implementation of heavy numerical computations. Typically Computer Audition 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 [115]. A top-down (prediction-driven) approach has been used for computational auditory scene analysis [31]. An extension to this approach with a hierarchical taxonomy of sound sources is proposed in [81]. 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.

This chapter is mainly about architectural design and implementation issues and should be skipped by readers who are mainly interested in ideas about Computer Audition algorithms. It should be useful for readers who are interested in implementing practical, real-time Computer Audition systems. Publications in Computer Audition rarely go into implementation details (in many cases because their performance it too slow). The result is that people entering the field have to discover the correct abstractions by trial and error.

It is my hope that this chapter and the software it describes will help people entering the field concentrate on new ideas and avoid rediscovering and reimplementing basic building blocks of Computer Audition research.

6.2 Related Work

This section will be a little bit different than the corresponding sections of the previous chapters in that it references software systems rather than publications. Software frameworks from other application areas that have similar design goals to Marsyas as well as alternatives to Marsyas for implementing Computer Audition algorithms will be described.

There are many software frameworks for a variety of areas in Computer Science. Representative examples (references can be found at the corresponding web pages) are:

- **Audio Synthesis**

  CSOUND, [1] Synthesis Toolkit (STK), [2], Pure Data (Pd) [3]

- **Image Processing and Manipulation**

  GIMP, [4] Khoros, [5]

- **Visualization**

  Visualization Toolkit (VTK), [6]

When designing and developing Computer Audition software there are several possible implementation choices each with their own advantages and disadvantages. A common

[1]`http://mitpress2.mit.edu/e-books/csound/frontpage.html`
[2]`http://www-ccrma.stanford.edu/software/stk/`
[3]`http://www.pure-data.org/`
[4]`http://www.gimp.org`
[5]`http://www.tnt.uni-hannover.de/js/soft/imgproc/khoros/`
[6]`http://public.kitware.com/VTK/`

approach is to use a numerical computation environment (typically MATLAB [7]). The main advantage of this approach are: 1) the large number of available numerical functions 2) simple syntax simplified for numerical computations 3) interactive computing and plotting. The main disadvantages are: 1) slow performance (MATLAB is memory intensive and slow compared to efficient C or C++ implementations) 2) lack of flexibility of a general programming language (although MATLAB fans might disagree) 3) difficult to create user interfaces that respond in real time to audio signals 4) expensive licence updates 5) not easy to port to new platforms. In many cases in order to overcome some of these problems a hybrid approach is used, where the critical parts are implemented in C or C++ and MATLAB is used for further processing and integration.

Another approach is to start from scratch using a general purpose programming language and provide a separate efficient implementation for each developed algorithm. The main disadvantage of this approach is the significant effort it requies because code has to be rewritten from scratch for every new algorithm. The use of standard libraries can provide some code reusability but does not allow easy addition of new components.

Another possibility is the use of an audio synthesis framework such as CSOUND, STK or PD. Although a lot of the required functionality such as reading soundfiles and playing audio is already there, these frameworks are not targeted for Computer Audition and therefore lack certain important components such as Machine Learning algorithms.

The approach chosen in this work is to develop a framework for Computer Audition research in a general programming language and try to abstract the basic building blocks and make it flexible so that new blocks can be added easily. The *Marsyas* software framework is the result of these efforts. It was first described in [124, 129] and has been used to design and develop all the described algorithms in this thesis as well as conduct the evaluation user experiments. The advantage of this approach is that, if done properly, it combines

[7]`http://www.mathworks.com`

expressivity and flexibility with performance. The main disadvantage is that it requires significant effort in design and implementation until it reaches a stage that it is usable.

6.3 Architecture

Marsyas follows 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. Another advantage is that the server or the client can be rewritten in some other programming language without needing to change the other part as long as the interface conventions between the two are respected.

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 extractor for Music/Speech classification to avoid calculating features on silent frames. Similarly hierarchical classification can be expressed using multiple extractors using different features.

6.3.1 Computation Server

The computation server contains the code for reading/writing/playing audio files, feature extraction, signal processing and machine learning. Special attention was given to abstracting these layers 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:

Systems process an input vector of floating point numbers and output another vector (of possibly different dimensionality). They encapsulate all the Signal Processing functionality of *Marsyas*. The parameters of a *System* and any temporary memory storage it might require are initialized and allocated when the object is constructed. The process method can then be applied, usually more than one time, to floating point vector objects of a specific input and output size specified at construction time. Example of Systems include transformations such as: windowing (for example Hamming), Fast Fourier Transform, and Wavelets. In addition features such as the Spectral Centroid, Rolloff and Flux are also represented as systems. Systems are similar to plugins or unit generators of audio processing software. Systems can be constructed from arbitrary flow networks of other systems using combinators. For example the whole computation of features for Music/Speech classification can be encapsulated in one system as shown in Figure 6.1.

Combinators provide ways to form complex *Systems* by combining simpler *Systems*. The *Parallel* combinator takes as input a list of *Systems* that have the same same input size and produces a system that takes the same input and has as output the concatenation of ouputs of the *Systems* it combines. The *Series* combinator takes as input a list of *Systems* and creates a system that applies them one after the other (the input and output sizes of adjacent pairs have to match). Figure 6.1 shows how Parallel and Series combinators are used to construct the flow network for Music/Speech classification.

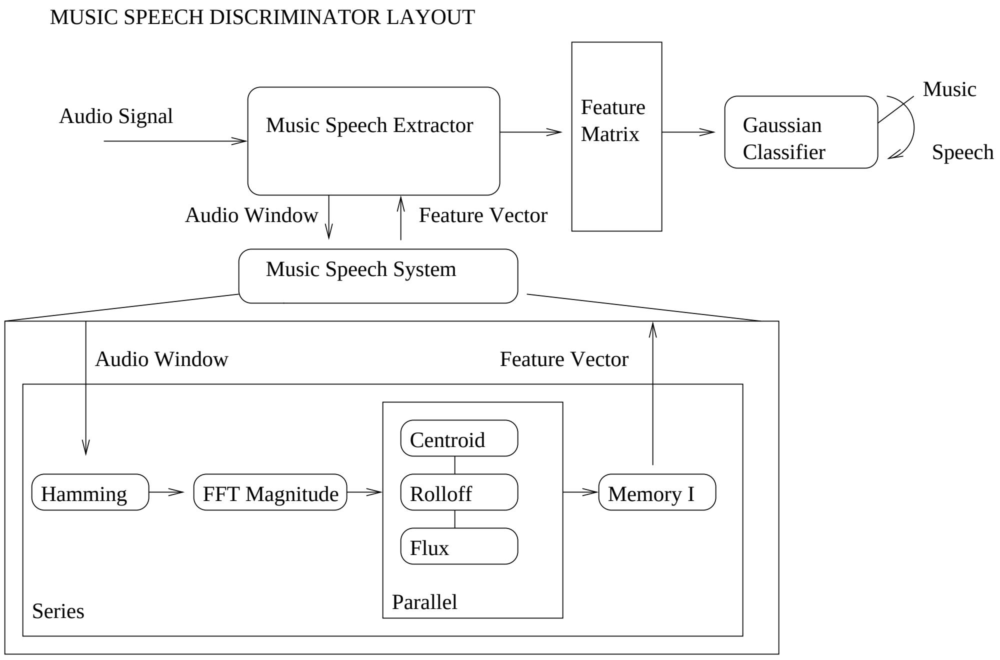


Figure 6.1: Music Speech Discriminator Architectural Layout

Memories are a particular case of *Systems* that correspond to 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.

Extractors break up a sound stream into frames. For each frame they use a complex *System* to calculate a *Feature Vector*. The result is a time-series of feature vectors which is called the *Feature Matrix*. Typically, there is a different extractor for each classification scheme.

Classifiers are trained using a *Feature Matrix* and can then be used to classify *Feature Vectors* or *Feature Matrices*. They also support various methods for debugging and evaluation such as k-fold cross-validation.

Segmentors take as input *Feature Matrix* and output a *Time Line* corresponding to the extracted segments.

Data structure objects:

Signals produce or consume sound segments represented as vectors of floating point numbers. Examples of signals include: various readers/writers of sound file formats, real-time audio IO, and synthesis algorithms.

Float Vectors-Matrices 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.

Feature Matrix stores automatically extracted audio features in the form of a *Feature Matrix*, and map them to a *Label Matrix* that is used for training, classification and segmentation. For example a particular *Feature Vector* might be mapped to two labels, one corresponding to the class it belongs (for example Jazz) and the other to the file that it originates from.

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 Figure 6.2). Trees have been shown to be an effective abstraction for browsing multimedia data [136].

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

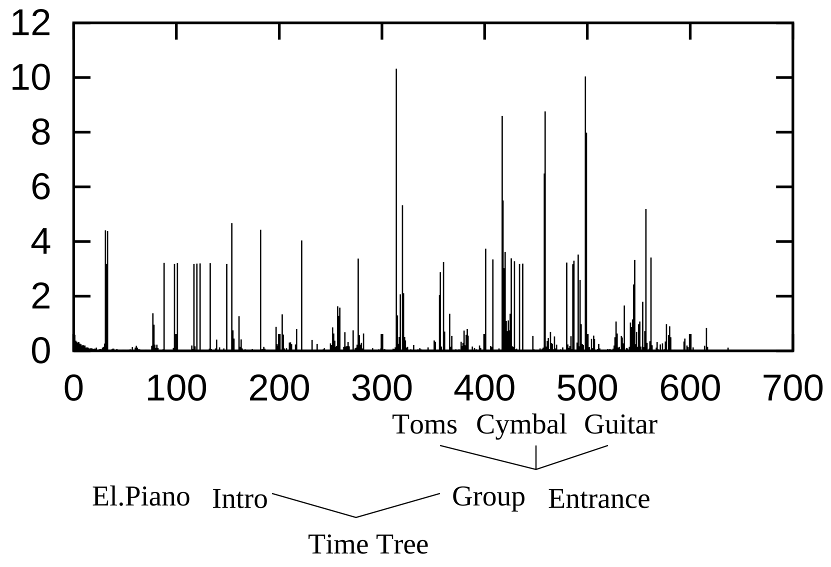


Figure 6.2: Segmentation peaks and corresponding time tree

6.3.2 User Interface Client - Interaction

The client contains all the code for the user interfaces and is written in Java. Because all of the performance critical code is executed in the server, the choice of Java does not impact performance and allows the use of the excellent JavaSwing graphical user interface framework. The main user interface components and their architecture were described in Chapter 4. In this section some of additional details about the architecture and implementation are provided.

All the interfaces follow a Model-View-Controller design pattern [64] that allows mutliple views of the same underlying data to be created. *Browsers* render audio collections based on parameters stored in *MapFiles*. In many cases the *MapFiles* are automatically generated using Computer Audition algorithms. Using *Browsers* the space can be explored and a subset of the collection can be selected (logical zooming) for subsequent browsing or editing. Editing the selected files can be done using *Viewers* and *Editors* which are also configured by *MapFiles*, possibly automatically generated. During playback of the

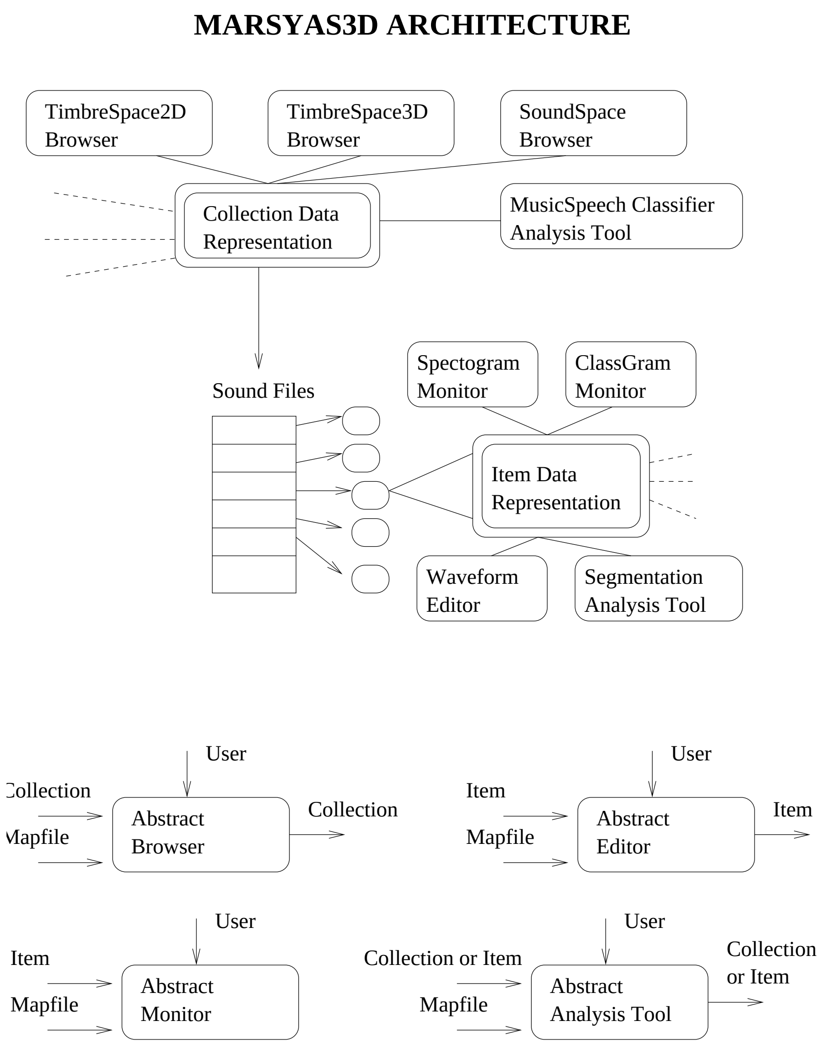


Figure 6.3: Marsyas User Interface Component Architecture

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 6.3.

6.4 Usage

Marsyas can be used in a variety of different levels depending on the user experience and flexibility required. At the highest level the graphical user interfaces can be used directly to browse, analyze and edit audio signals and collections. At a lower level, command line tools can be used to perform basic Computer Audition operations. For example the following sequence of commands extracts Spectral Shape features based on the STFT, trains a music/speech Gaussian classifier and evaluates the results. The output will be the classification accuracy using 10-fold cross-validation and the corresponding confusion matrix.

```
mkcollection music.mf /home/gtzan/data/sound/music/
mkcollection speech.mf /home/gtzan/data/sound/speech/
bextract FFT ms music.mf speech.mf
evaluate 0 GS ms 100
```

If more flexibility is required, complex Computer Audition algorithms can be constructed by combining various objects that correspond to basic building blocks. Many of the projects done by undergraduate students using Marsyas are done this way without having to write any code other than the glue that connects the building blocks. Finally, arbitrary new algorithms can be added by writing source code directly and extending via inheritence the existing abstract classes of the system.

6.5 Characteristics

In addition to the described components, for the purposes of the MPEG-based feature calculation, an MPEG audio decoder was implemented in C++. The source code is loosely based on the Fraunhofer institute reference software implementation and other open source implementations [8]. The combined decoding and classification/segmentation runs real-time on a Pentium II PC.

[8] `http://www.mpeg.org`

All the graphical user interface components are implemented in Java. The Java3D API is used for the 3D graphics and JavaSound API is used for the audio and MIDI playback. The computation server is written in standard ANSI C++. For parametric synthesis of sound effects and audio generation from MIDI the Synthesis Toolkit [9] [23] is used. MARSYAS is mainly developed under Linux and has also been compiled under Solaris, Irix, Mac OS X, and Windows (Visual Studio and Cygwin compilers) operating systems. The use of standard C++ and JAVA and the independence from external libraries makes the system easily portable to new architectures. Moreover, the client-server architecture allows a variety of possible configurations. For example, it is possible to have Windows and Solaris clients accessing the audio information retrieval server running in Linux. The framework is available as free software under the GNU Public Licence (GPL) from: `http://www.cs.princeton.edu/~gtzan/marsyas.html` and has been downloaded approximately 2500 times from 1000 different hosts. *Marsyas* has been used by many Princeton undergraduate and graduate students doing projects related to Computer Audition. Some examples are: Music Mosaics (Douglas Turnbull, Ari Lazier), Instrument classification (Corrie Elder), Pitch Histograms (George Tourtellot), query user interfaces (Andreye Ermolinskyi), and Music Library (John Forsyth).

[9] `http://ccrma-www.stanford.edu/software/stk/`

Chapter 7

Conclusions

Remember, Information is not knowledge, Knowledge is not Wisdom; Wisdom is not truth; Truth is not beauty; Beauty is not love; Love is not music; Music is the best.

Frank Zappa

7.1 Discussion

In this thesis, a variety of algorithms, techniques and interfaces for working with audio signals and collections were proposed, developed, and evaluated. These systems can be used for manipulation, analysis and retrieval of audio signals and collections. Some of the main characteristics they share are: 1) practical solutions to fundamental problems that are applicable to arbitrary audio signals 2) utilization of large collections 3) statistical evaluation in many cases based on user studies 4) use of visualization and human computer interaction techniques to include the user in the system 5) integration under a common software implementation. These characteristics can be summarized as a general tendency to create practical workable systems that can be used as prototypes for the *Computer Audition* systems of the future.

To summarize the main contributions of this thesis are: 1) calculation of features from MPEG audio compressed data 2) calculation of features based on the DWT 3) a general multifeature methodology for segmentation of arbitrary textures 4) automatic musical genre classification and similarity retrieval based on timbral, rhythmic and harmonic features 5) a variety of novel user interfaces such as Timbregrams and Timbrespace for interacting with audio signals and collections and for specifying audio queries such as Soundsliders 6) the creation of an extensible software framework for Computer Audition research that integrates the proposed systems as well as a large variety of existing algorithms and systems.

This is a good point to go back to Section 1.9 and read again the proposed Computer Audition usage scenarios and see how the described systems in this thesis serve their requirements. Our hope is that someday in the future, the tools for interacting with audio signals and collections will be inspired by and look like our prototype systems. In the following Sections current and future work related to this thesis will be described.

7.2 Current Work

In this Section some of the ongoing work, continuing the research described in this thesis, will be presented. The *Sonic Browser* is a tool for accessing sounds or collections of sounds using sound spatialization and content-overview visualization techniques developed at the University of Limerick, Ireland. We are currently collaborating with the *Sonic Browser* team, combining their system with *Marsyas*. More specifically the new system, under development, combines the interactivity and user configuration of the *Sonic Browser* with the automatic audio information retrieval tools of *Marsyas* in order to create a flexible distributed graphical user interface with multiple visualization components that provides novel ways of browsing large audio collections. The Extensible Markup Language (XML) is used to exchange configuration and metadata information between the various developed components. This work has been submitted to the International Conference for Auditory Display (ICAD) 2002.

In Chapter 5 it was shown that features calculated from Pitch Histograms can be used for automatic musical genre classification. However, for the cases that these features fail it is not clear if these failures have to do with the choice of features or with errors in the multiple pitch detection algorithm. In order to address this question we have compared the performance of Pitch Histogram features calculated on MIDI data and on audio derived from the same MIDI files. By definition, Pitch Histograms derived from MIDI data will have no pitch errors therefore by comparing the two results we can judge to what extent pitch errors influence the use of features for classification. It has been shown that the Pitch Histograms features in both cases show the same performance therefore it can be concluded that the possible pitch errors in the audio cases do not significantly affect the classification performance of the features. Of course, audio from MIDI signals although polyphonic are easier than “real world” audio signals for multiple pitch detection. However, because the genre classification results obtained from MIDI and audio-from-MIDI are comparable to results obtained from “real-world” audio data for similar tasks (but obviously different datasets) it can be concluded that the classification performance of the features in “real world” audio data is also not significantly affected by pitch detection errors. It is important to note that this does not mean that better features are not possible or that the multiple pitch detection algorithm is perfect. This work will be submitted to the International Symposium on Music Information Retrieval (ISMIR) 2002.

As already mentioned, in addition to digital music distribution, collections sound effects are another important area for Computer Audition research. The perceptual similarity of sound effects has been investigated by conducting user experiments in [104]. We are collaborating with the people involved in this project to use the described tools and interfaces to visualize and compare the results of this work. In addition these results will be compared with automatically extracted feature vectors and similarity information over the same collection of digital sound effects.

7.3 Future work

Computer Audition is a relatively recent research area and there are a lot of interesting directions for future work. Some of these directions are obvious extensions of the work described in this thesis and others are more unrelated. It is our hope, that the developed software framework will help us and other researchers realize some of these directions for future work.

In the representation stage, investigation of different front-ends for feature extraction is an area of active research. Although most of the features proposed in the literature have shown adaquate performance, there have been few thorough investigations of all possible feature choices and combinations. By providing a common framework and data collections we are planning to compare different feature front-ends and optimize the choice of parameters for particular applications. For example we are investigating the use of computational models of the ear physiology such as the ones proposed in [116, 117] as the basis for feature extraction.

Although there are have been a large number of studies about the perception of beat periodicities there is relatively little work in the perception of beat strength. We are currently conducting a user experiment to investigate how humans perceive musical beat strength. Each subject has to rate 50 segments from a variety of musical styles into five beat strength groups (weak, medium-weak, medium, medium-strong, strong). The order of presentation is randomized to avoid any order effects such as learning. Preliminary results from the 15 subjects that have completed the study so far indicate that there is considerable agreeement between subjects reagrding music beat strength. the results of this study will be used to evaluate and calibrate the calculation of Beat Histograms.

Another interesting direction is the use of compressed domain audio features for other types of audio analysis than music/speech classification, instrument classification and segmentation. Some examples are multiple pitch detection, beat analysis, speaker identification and speech recognition. The features described in this work are not the only ones possible and further investigation is needed in order to come up with better features tailored to specific applications. In addition to the subband analysis other types of information can be utilized. For example in MPEG layer III a window switching scheme is used where the 32 subband signals are further subdivided in frequency content by applying, to each subband, a 6-point or 18-point modified DCT block transform. The 18-point block transform is normally applied because it provides better frequency resolution, whereas the 6-point block transform provides better time resolution. Therefore it is more likely to find semgentation boundaries in short blocks and this information could be used to enhance the segmentation algorithm. For statistical pattern recognition large amounts of training data need to be collected. By doing the analysis on MPEG compressed data the enormous resources available on the Web can become available for training. We are planning to implement a Web crawler to automatically gather .mp3 files from the Web for this purpose.

The exact details of feature calculation for older methods such as the STFT and MFCC have been relatively well explored in the literature. On the other hand, features based on the DWT have appeared only recently. Further improvements in classification accuracy can be expected with more careful exprimentation with the exact details of the parameters. We also plan to investigate the use of other wavelet families for the feature computation. Another interesting direction is to combine features from different analysis techniques to improve classification accuracy. We aslo plan to apply the general methodology for audio segmentation based on “texture” decribed in this thesis, using the DWT-based features.

In the implementation of the segmentation algorithm the importance of each feature is equal. Most likely this is not the case with humans and is not optimal. Therefore, relative weighting of the features as well as optimization of the parameters and heuristics need to be explored. One possible approach for this problem of feature weighting is the use of Genetic algorithms as in [44]. The use of the proposed segmentation methodology for other types of audio signals such as news broadcasts is also an interesting possibility. Because of the difficulty of evaluating and comparing different segmentation schemes standard corpora of audio examples need to be designed. Standarized corpora are very important for Computer Audition research and unfortunately their creation has been hindered because of copyright problems. In the future we plan to collect more data on the time required to segment audio. Empirical evidence is required that the automatic segmentation reduces the time to segment. Further tests in thumbnailing will need to be devised to determine the salience of the human and machine selected thumbnails, and to determine their usefulness. For example, can thumbnails cause a speedup in location/indexing, or can thumbnails be concatenated or otherwise combined to construct a useful ”caricature” of a long audio selection? We plan to make more detailed analyses of the text annotations. Finally the developed graphical user interface allows the collection of many user statistics like number of edit operations, detailed timing information,etc. that we plan to investigate further.

For the beat analysis using Beat Histograms a comparison of the beat detection algorithm described in [108] with our scheme on the same dataset is also planned for the future. In addition, more careful studies using the synthetic examples are also planned. These studies will show if it is possible to detect the phase and periodicity of multiple rhythmic lines using the DWT.

In this thesis similarity retrieval was performed using a single feature vector representation for each file. Although this approach works well for files with relatively constant texture it has problem with files such as classical music with many different sections. Segmentation algorithms can provide information about where texture changes occur but the problem of how to do similarity retrieval in those cases still remains. In the future we plan to investigate trajectory and segmentation-based approaches to content-based similarity retrieval.

For the automatic musical genre classification an obvious direction for future research is expanding the genre hierarchy both in width and depth. Other semantic descriptions such as emotion or voice style will be investigated as possible classification categories. More exploration of the pitch content feature set could possibly lead to better performance. Alternative multiple pitch detection algorithms, for example based on cochlear models, could be used to create the pitch histograms. For the calculation of the beat histogram we plan to explore other filterbank front-ends as well as onset based periodicity detection as in [66, 112]. We are also planning to investigate real-time running versions of the rhythmic structure and harmonic content feature sets. Another interesting possibility is the extraction of similar features directly from MPEG audio compressed data as in [95, 130]. Finally, we are also planning to use the proposed feature sets with alternative classification and clustering methods such as artificial neural networks. Two other possible sources of information about musical genre content are melody and singer voice. Although melody extraction is a hard problem that is not solved for general audio it might be possible to obtain some statistical information even from imperfect melody extraction algorithms. Singing voice extraction and analysis is another interesting direction for future research.

In the Interaction stage, there is a lot of work that has to be done in order to improve the described interfaces and evaluate their effectiveness. For example, based on the fact that on average the subjects needed more than 2 hours for segmenting 10 minutes of audio using standard audio editing tools we plan to compare how much this process can be accelerated using the Enhanced Audio Editor. By using our own software, collecting usage statistics such as mouse clicks and timing information is much easier than using a commercial audio editor. Some other directions for future work are the development of alternative sound-analysis visualization tools and the development of new model-based controllers for sound synthesis based on the same principles. Of course new analysis and synthesis algorithms will require adjustment of our existing tools and development of new ones. More formal evaluations of the developed tools are planned for the future. Integrating the model-based controllers into a real virtual environment is another direction of future work.

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.

It is our belief that the problem of query specification has many interesting directions for future research because it provides a new perspective and design constraints to the issue of automatic music and sound generation and representation. Unlike existing work in automatic musical generation where the goal is to create as good music as possible, the goal in our work is to create convincing sketches or caricatures of the music that can provide feedback and be used for music information retrieval. Another related issue is the visual design of interfaces and displays and its connection to visual music and score representations.

Evaluating a complex retrieval system with a large number of graphical user interfaces is difficult and can only be done by conducting user studies. For the future we plan a task-based evaluation of our system similar to [57] where users will be given a specific task such as locating a particular song in a large collection and their performance using different combinations of tools will be measured.

Another direction for future research is the collaboration with composers and music cognition experts with the goal of exploring different metaphors for sketching music queries for users with and without musical background and how those differ. Currently a lot of information about audio and especially music signals can be found in the form of metadata annotations such as filenames and ID3 tags of mp3 files. We plan to develop web crawlers that will collect that information and use it to build more effective tools for interacting with large audio collections.

On the implementation side, having a well-designed framework and knowing the basic abstractions and building blocks makes the porting to other programming languages and systems easier. Currently versions of the computational server written in SMLNJ (a typed functional programming language) and JAVA are under development. A version of some of the client user interfaces written, using GTK, is under development.

7.3.1 Other directions

In this Section some directions for future work that are not direct extensions of the work described in this thesis will be described. As already mentioned, a large number of existing Computer Audition research solves hard problems on small collections of limited "toy" examples with the hope of gradually increasing the number and complexity of signals it can handle. This approach has not yet been successful in building practical systems. In contrast, the work described in this thesis emphasizes simpler, fundamental problems that can be reliably solved over large collections of "real-world" complex audio signals. Therefore an obvious directions for future work is to try to solve increasingly harder problems while still keeping the methods applicable to "real world" audio signals. These harder problems can be seen as intermediate steps toward the goal of automatic polyphonic transcription. In a sense, some of the techniques described in this work such as Beat Histograms and Pitch Histograms are moving toward that goal. In the following paragraphs a number of such interesting unsolved problems are described.

An important source of information about musical signals is the number and type of instruments playing at any particular moment (for example a saxophone, piano and drums are playing now). Instrument mixture identification refers to automatically solving this problem. Although it is a harder problem than classification as there has to be some separation of the signals, it is an easier problem than polyphonic transcription or source separation as only the instrument labels are required. A related problem is the problem of instrument tracking where a statistical model of an instrument sound is provided and that particular instrument is tracked through a polyphonic recording. Another interesting problem is the separation of particular types of sounds or instruments such as bass, drums and voice. For example the separation of drums sounds could be used for more elaborate detection of particular rhythmic styles such as Bossa Nova.

In the same manner that Pitch Histograms provide overall pitch information for a song even though multiple pitch detection is still not perfect, it is probably possible to detect and identify chords. Beat analysis method can also provide metric information as well as higher level structural information. An important source of information for identifying music is the singing voice. Therefore techniques for singer identification and singing voice separation or enhancement are very interesting.

In addition to applications related to audio signals, it is possible that many of the proposed ideas have applications in other areas and types of signals. Some obvious examples include video and speech signals and some less obvious include measured experimental data from fields such as physics and astronomy, stock market information, and biological signals. For example, visualizations of DNA based on Signal Processing techniques similar to the ones described in this thesis, are described in [6].

7.4 Final Thoughts

Audio signals and especially musical signals are very interesting from a philosophical perspective. For some reason, humans find organized structures of moving air molecules so interesting that they have sophisticated sensory organs for detecting them, spend significant amounts of their lives listening to them, make weird devices for creating them, and use sophisticated technology to store and reproduce them. In the early days of humanity, the construction of practical tools such as arrows and knives was paralleled by the creation of holed tubes and stretched membranes for setting air molecules in motion. Today the construction of practical tools such as operating systems and firewalls is paralleled by the creation of systems for helping humans interact and explore the vast amounts of structured air molecule motions stored as strings of ones and zeros. Music is the best.

Appendix A

A.1 Introduction

This appendix is a brief overview of some standard techniques that are important for the work described in this thesis. The main goal is to describe how the algorithms work rather than explaining and proving why they work. More details can be found in the corresponding references.

A.2 Short Time Fourier Transform

The most common set of features used for classification, segmentation and retrieval are based on the FFT and more specifically the Short Time Fourier Transform (STFT). The STFT $X(n,k)$ of a signal $x(n)$ is a function of both time n and frequency k. It can be written as as:

$$X(n,k) = \sum_{m=-\infty}^{\infty} x(m)h(n-m)e^{-j(2\pi/N)km} \tag{A.1}$$

where $h(n)$ is a sliding window, $x(m)$ is the input signal, and N is the size of the transform. Typically the STFT is implemented using the FFT algorithm to make its calculation computationally efficient. The STFT can be viewed as a filter bank where the kth filter channel is obtained by multiplying the input $x(m)$ by a complex sinusoid at frequency k/N times the sample rate and then convolving with the low pass filter $h(m)$. In other words, the output $X(n,k)$ for any particular value of k, is a frequency shifted, band-pass filtered version of the input. Another way to view the STFT, is that for any particular value of n, the STFT is the Discrete Fourier Transform of the windowed input at time n. In other words, the STFT can also be thought as partially overlapping Fourier transforms. The STFT for a particular frequency k at particular time n is a complex number. Typically for feature calculation only the magnitude of these complex numbers is retained.

A.3 Discrete Wavelet Transform

In the pyramidal algorithm the signal is analyzed at different frequency bands with different resolutions for each band. This is achieved by successively decomposing the signal into a coarse approximation and detail information. The coarse approximation is then further decomposed using the same wavelet decomposition step. This decomposition step is achieved by successive highpass and lowpass filtering of the time domain signal and is defined by the following equations:

$$y_{high}[k] = \sum_n x[n]g[2k-n] \tag{A.2}$$

$$y_{low}[k] = \sum_n x[n]h[2k-n] \tag{A.3}$$

where $y_{high}[k]$, $y_{low}[k]$ are the outputs of the highpass (g) and lowpass (h) filters, respectively after subsampling by two. Each level of the pyramid corresponds roughly to frequency bands spaced in octaves. Because of the symmetric pyramid structure and the decimation the DWT can be performed in $(O(N)$ (where N is the number of input points). In this work, the DAUB4 filters proposed by Daubechies [25] are used. Figure A.1 shows a schematic diagram of the pyramidal algorithm for the calculation of the Discrete Wavelet Transform.

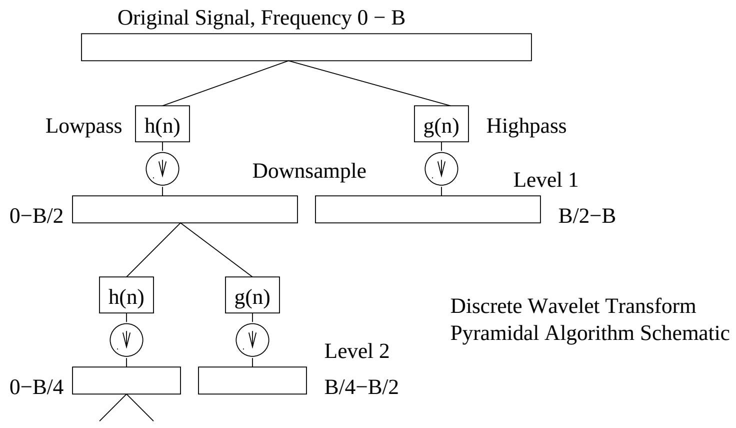

Original Signal, Frequency 0 − B
Lowpass
h(n)
g(n)
Highpass
Downsample
Level 1
0−B/2
B/2−B
h(n)
g(n)
Discrete Wavelet Transform
Pyramidal Algorithm Schematic
Level 2
0−B/4
B/4−B/2

Figure A.1: Discrete Wavelet Transform Pyramidal Algorithm Schematic

A.4 Mel-Frequency Cepstral Coefficients

Mel-Frequency Cepstral Coefficients [26] are a common feature front-end that is used in many speech recognition systems. An exploration of their use in Music/Speech classification is described in [72]. This technique combines an auditory filter-bank, followed by a cosine transform to give a representation that has roughly properties similar to the human auditory system.

More specifically the following steps are taken in order to calculate MFCCs:

1. The short-time slice of audio data to be processed is windowed by a Hamming window
2. The magnitude of the Discrete Fourier Transform is computed using the FFT algorithm
3. The FFT bins are combined into filterbank outputs approximating the perceptual frequency resolution properties of the human ear. The filterbank is constructed using

13 linearly-spaced filter (133.33 KHz between their center frequencies) followed by 27 log-spaced filters (separated by a factor of 1.0711703 in a frequency). . The bins are summed using a triangular weighting function which starts risting at the CF of the previous filter, has a peak at the current filter CF, and ends at the CF of the next filter. (see Figure A.2).

4. The filterbank outputs are converted to log base 10
5. The Discrete Cosine Transform of the outputs is used to reduce dimensionality.

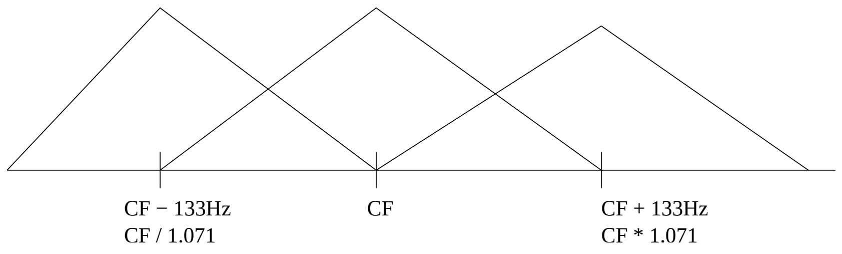


Figure A.2: Summing of FFT bins for MFCC filterbank calculation

More details about the calculation of MFCCs can be found in [98].

A.5 Multiple Pitch Detection

The multiple pitch detection used for Pitch Histogram calculation, is based on the two channel pitch analysis model described in [123]. A block diagram of this model is shown in Figure A.3. The signal is separated into two channels, below and above 1 kHz. The channel separation is done with filters that have 12 dB/octave attenuation at the stop band. The lowpass block also includes a highpass rolloff with 12 dB /octave below 70Hz. The high-channel is half-wave rectified and lowpass filtered with a similar filter (including the highpass characteristic at 70Hz) to that used for separating the low channel.

The periodicity detection is based on "generalized autocorrelation" i.e the computation consists of a discrete Fourier transform (DFT), magnitude compression of the spectral representation, and an inverse transform (IDFT). The signal x2 of Figure A.3 is obtained as:

$$x_2 = IDFT(|DFT(x_{low})|^k) + IDFT(|DFT(x_{high})|^k) \tag{A.4}$$

$$= IDFT(|DFT(x_{low})|^k + DFT(x_{high})|^k) \tag{A.5}$$

where x_{low} and x_{high} are the low and high channel signals before the periodicity detection blocks in Figure A.3. The parameter k determines the frequency domain compression. For normal autocorrelation $k = 2$. The fast Fourier Transform (FFT) and its inverse (IFFT) are used to speed the computation of the transforms.

The peaks of the summary autocorrelation function (SACF) (signal $x2$ of Figure A.3 are relatively good indicators of potential pitch periods in the signal analyzed. In order to filter out integer multiple of the fundamental period a peak pruning technique is used. The original SACF curve, is first clipped to positive values and then time-scaled by a factor of two and subtracted from the original clipped SACF function, and again the result is clipped to have positive values only. That way, repetitive peaks with double the time lag of the basic peak are removed. The resulting function is called the enhanced summary autocorrelation (ESACF) and its prominent peaks are accumulated in the Pitch Histogram calculation. More details about the calculation steps of this multiple pitch detection algorithm, as well as its evaluation and justification can be found in [123].

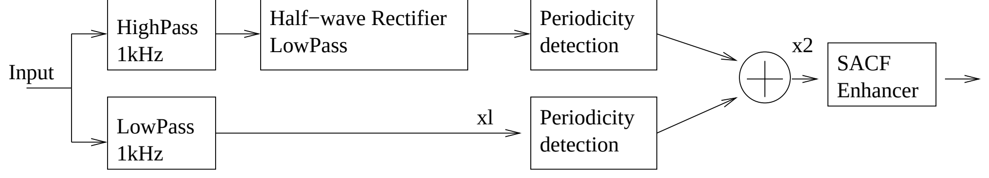


Figure A.3: Multiple Pitch Analysis Block Diagram

A.6 Gaussian classifier

The Gaussian classifier models each class as a multidimensional Gaussian distribution. The Gaussian or Normal distribution in d dimesion can be written as:

$$p(\mathbf{x}) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}}^{1} \exp\left[-\frac{1}{2}(\mathbf{x}-\mu)\Sigma^{-1}(\mathbf{x}-\mu)\right] \tag{A.6}$$

where x is a d-component column vector, μ is the d-component mean vector, Σ is the d-by-d covariance matrix, and $|\Sigma|$ and Σ^{-1} are its determinant and inverse respectively.

A Gaussian classifier can be fully characterized by its mean vector μ and covariance matrix Σ. Maximum Likelihood methods view the parameters of a distribution as quantities whose values are fixed but unknown. The best estimate of their values is defined to be the one that maximizes the probability of obtaining the samples actually observed. It can be shown that the Maximum Likelihood estimate of parameters for the Gaussian classifier is the sample mean and covariance matrix [30]. In other words, in order to train a Gaussian classifier for a particular class of labelled samples, the sample mean and covariance matrix are used as the parameters of the distribution. Classification of an unknown vector x using a Gaussian classifier is done by finding the class Gaussian distribution that most likely would produce this vector.

A.7 GMM classifier and the EM algorithm

The GMM classifier models each class probability density function as a finite mixture of multidimensional Gaussian distributions. If we denote θ_m the parameters of each Gaussian distribution mixture component and θ the complete set of parameters $\theta_1, ..., \theta_k, \alpha_1, ...\alpha_k$ then the probability density function of the mixture model can be written as:

$$p(\mathbf{y}|\theta) = \sum_{m=1}^{k} \alpha_m p(\mathbf{y}|\theta_m) \tag{A.7}$$

where the α_m are the mixing probabilities. Given a set of n independent and identically distributed samples $Y = \mathbf{y}^{(1)}, \mathbf{y}^{(2)}, ..., \mathbf{y}^{(n)}$, the log-likelihood corresponding to a k-component mixture is

$$\log p(Y|\theta) = \log \prod_{i=1}^{n} p(\mathbf{y}^{(i)}|\theta) = \sum_{i=1}^{n} \log \sum_{m=1}^{k} \alpha_m p(\mathbf{y}^{(i)}|\theta_m) \tag{A.8}$$

Our goal is to find the maximum likelihood (ML) estimate:

$$\theta_{ML} = \arg\max_{\theta} \{\log p(Y|\theta)\} \tag{A.9}$$

It is known that this estimate can not be found analytically therefore the EM algorithm which is an iterative procedure for finding local maxima of $\log p(Y|\theta)$ is used.

The EM algorithm is based on the intepretation of Y as incomplete data. For finite mixtures, the missing part is a set of n labels $Z = \mathbf{z}^{(1)}, \mathbf{z}^{(2)}, ..., \mathbf{z}^{(n)}$ associated with the n samples, indicating which component produced each sample. Each label is a binary indicator vector. The complete log-likelihood (i.e., the one from which we could estimate θ if the complete data $X = \{Y, Z\}$ was observed) is

$$\log p(Y, Z|\theta) = \sum_{i=1}^{n} \sum_{m=1}^{k} z_m^{(i)} \log \left[\alpha_m p(\mathbf{y}^{(i)}|\theta_m)\right] \tag{A.10}$$

The EM algorithm produces a sequence of parameter estimates $\{\theta(t), t = 0, 1, 2, ...)\}$ by alternatingly applying two steps (until some convergence criterion is met):

- **E-step**: Computes the conditional expectation of the complete log-likelihood, given Y and the current estimate of the parameters $\theta(t)$.

$$Q(\theta, \theta(t)) = E\left[\log p(Y, Z|\theta)|Y, \theta(t)\right] \tag{A.11}$$

- **M-step**: Updates the parameter estiamtes according to:

$$\theta(t+1) = \arg\max_{\theta} Q(\theta, \theta(t)) \tag{A.12}$$

In our implementation, the K-Means algorithm described in the next Section is used to initialize the means of each component and diagonal covariance matrices are used. More details about the EM algorithm can be found in [84].

A.8 K-Means

The K-Means algorithm is an unsupervised learning or clustering algorithm. The only input parameter is the desired number of clusters and the output of the algorithm is the mean vectors $\mu_1, \mu_2, ..., \mu_n$ of each cluster. The algorithm is iterative and has the following steps:

1. Initialize n,c, $\mu_1, \mu_2, ..., \mu_n$
2. Do: Classify n samples according to nearest μ_i
3. Recompute μ_i
4. Until no change in μ_i
5. Return $\mu_1, \mu_2, ..., \mu_n$

More details and variations can be found in [30].

A.9 KNN classifier

The K nearest neighbors classifier (KNN) is an example of a non-parametric classifier. If we denote $D^n = \mathbf{x}_1, ...\mathbf{x}_n$ a set of n labeled prototypes then the nearest neighbor rule for classifying an unknown vector x is to assign it the label of its closest point in the set of labeled prototypes D^n. It can be shown that for an unlimited number of prototypes the error rate of this classifier is never worse than twice the optimal Bayes rate [30]. The KNN rule classifies x by assigning the label most frequently represented among the k nearest samples. Typically k is odd to avoid ties in voting. Because this algorithm has heavy time and space requirements various methods are used to make its computation faster and storage requirements smaller [30].

A.10 Principal Component Analysis

Principal Component Analysis (PCA) is a technique for dimensionality reduction [56]. PCA transforms an original set of variables into a new set of principal components which are at right angles (orthogonal) to each other (i.e uncorrelated). Basically the extraction of a principal component amounts to a variance maximizing rotation of the original variable space. In other words, the first principal component is the axis passing through the centroid of the feature vectors that has the maximum variance, therefore it explains a large part of the underlying feature structure. The next principal component tries to maximize the variance not explained by the first. In this manner, consecutive orthogonal components are extracted and each one describes an increasingly smaller portion of the variance. A schematic representation of Principal Component extraction is shown in Figure A.4.

If we denote the feature covariance matrix as:

$$E(xx^T) = \Sigma \tag{A.13}$$

then the eigenvectors corresponding to the N largest eigenvalues of Σ are the N principal component axes.

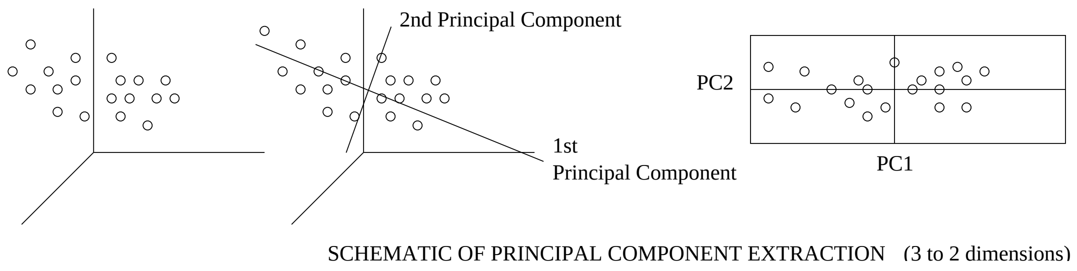

2nd Principal Component
1st
Principal Component
PC2
PC1
SCHEMATIC OF PRINCIPAL COMPONENT EXTRACTION (3 to 2 dimensions)

Figure A.4: Schematic of Principal Component extraction

Appendix B

B.1 Glossary

This section is a glossay of the basic terminology used in this thesis provided for quick reference. In most cases the meaning of those terms corresponds to their common use in the existing literature but in some cases (mainly interaction and implementation related) they are somewhat idiomatic. Because of the short nature of the definition some details that do not affect the use of the term in this thesis are left out. For more complete and correct definition the reader should consult the appropriate literature.

Analysis windows: short segments of sound with presumed relatively stable characteristics used to analyze audio content.

Beat: a relatively regular sequence of pulses that coincides with significant musical events

Beat tracking: is the process of extracting and following in real-time the main beat of a piece of music

Browser: an interface for interacting with a collection of audio files

Classification: the process of automatically assigning a categorical label to an audio signal from a list of predefined categories.

Clustering: the process of forming groups (clusters) of audio signals based on their similarity.

Collection: a list of audio files (typical size $>$ 100 items)

Computer Audition: the extraction of information from audio signals (as in Computer Vision).

Editor: an interface for interacting with a specific audio file

Feature Vector: a fixed-length array of real numbers used to represent the content of a short segment of sound called the analysis window. A feature vector can be thought of as a point in a multidimensional Feature Space.

Feature Space: A multidimensional space that contains Feature Vectors as points.

Fundamental frequency: is the main frequency of oscillation present in a sound

Mapping: a particular setting of interface parameters

Monitor: a display that is updated in real-time in response to the audio that is being played

MIDI: an interface specification for controlling digital music instrument. A file format (similar to a musical score) that stores control control information for playing a piece of music using digital music instruments

MPEG audio compression: Motion Pictures Expert Group audio compression is a perceptually based lossy audio compression scheme. The well known mp3 files are stored using this compression format

Mp3 files: audio signals compressed using MPEG audio compression

Musical Score: a symbolic representation of music that consists of the start, duration, volume and instrument quality of every note in a piece. Typically musical scores are represented graphically

Note: a sound of a particular pitch

Octave: the relation between to sounds that have a frequency ratio of two

Pitch: is a subjective property of sound that can be used to order sounds from low to high and is typically related to the fundamental frequency

Query User Interfaces: generate and/or filter audio signals in order to specify a audio retrieval query

Segmentation: the process of detecting changes of Sound Texture.

Selection: a list of selected audio files in a collection or a specific region in time in an audio file

Similarity retrieval: the process of detecting similar files to a query and returning them ranked by their similarity

Sound Texture: a particular type of sound mixture that can be characterized by its overall sound and audio feature statistics

Texture windows: larger windows used to characterize sound “texture” by collecting statistics of the “analysis windows”.

Timeline: a particular segmentation of an audio file to non-overlapping regions in time

Thumbnailing: the process of creating a short sound segment that is representative of a longer audio signal

View: a particular way of viewing a specific collection

Appendix C

C.1 Publications

The various topics and ideas in this thesis were presented in thematic order. The careful reader might have observed, while reading the evaluation chapter, that certain experiments have not been performed as a consequence of the chronological order of publications. For example in evaluating similarity retrieval a subset of musical content features were used because at the time the Beat Histogram and Pitch Histogram features had not been designed. This appendix provides a list of publications that this thesis is based on in chronological order with a short description of their content.

1. **A framework for audio analysis based on temporal segmentation and classification** 1999 [124]

   First description of the Marsyas software framework. Initial proposal for segmentation algorithm and implementation of Music/Speech classification based on [110] using the framework.

2. **Multifeature audio segmentation for browsing and annotation** 1999 [125]

   Full description of multifeature segmentation methodology and proposal of feature set. First user study for evaluate segmentation algorithm

3. **Experiments in computer-assisted annotation of audio** 2000 [128]

   Second user study for evaluating segmentation. Annotation and thumbnailing results collected. The experiments were conducted with the help of the developed Enhance Audio Editor user interface.

4. **Sound analysis using MPEG compressed audio** 2000 [130]

   Proposal of MPEG-based compressed domain audio features. Evaluation of the features based on Music/Speech classification and segmentation.

5. **Marsyas: a framework for audio analysis** 2000 [129]

   An enhanced publication in a Jouranl of [124].

6. **Audio Information Retrieval (AIR) Tools** 2000 [127]

   A high level overview of audio information retrieval. First description of Timbregrams and similarity retrieval evaluation study.

7. **3D Graphics Tools for Isolated Sound Collections** 2000 [126]

   Description of 3D graphics user interfaces. Introduction of Timbrespaces and sound effect query generators (PuCola, Gear, etc).

8. **MARSYA3D: A prototype audio browser-editor using a large scale immersive visual and audio display** 2001 [131]

   Description of graphics user interfaces and architecture. Emphasis on use of interface on the Princeton Display Wall.

9. **Audio Analysis using the Discrete Wavelet Transform** 2001 [134] Investigation of the Discrete Wavelet Transform for audio classification and beat analysis. Beat histograms are introduced.

10. **Automatic Musical Genre Classification of Audio Signals** 2001 [135] First paper on musical genre classification. Use of spectral features and beat histograms for classification.

11. **Audio Information Retrieval using Marsyas** 2002 [132] Chapter in book. High level overview of Marsyas, essentially a short summary of this thesis.

12. **Musical Genre Classification** 2002 [133] Detailed experiments in musical genre classification. Pitch Histograms introduced and used together with spectral features and beat histograms features for classification.

Appendix D

D.1 Web Resources

Web resources related to the research presented in this thesis:

- `http://soundlab.cs.princetone.edu/demo.php`

  Demonstrations

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

  The Marsyas software framework.

- `http://www.cs.princeton.edu/˜gtzan/publications.html`

  My Publications.

- `http://www.cs.princeton.edu/˜gtzan/caudition.html`

  Lots of web resources (people, conferences, companies, publications, etc)

Bibliography

[1] *Proc. Int. Symposium on Music Information Retrieval (ISMIR)*, Plymouth, MA, 2000.

[2] *Proc. Int. Symposium on Music Information Retrieval (ISMIR)*, 2001.

[3] M. Alghoniemy and A. Tewfik. Rhythm and Periodicity Detection in Polyphonic Music. In *Proc. 3rd Workshop on Multimedia Signal Processing*, pages 185–190, Denmark, Sept. 1999.

[4] M. Alghoniemy and A. Tewfik. Personalized Music Distribution. In *Proc. Int. Conf. on Acoustics, Speech and Signal Processing ICASSP*, Istanbul, turkey, June 2000. IEEE.

[5] E. Allamanche, H. Jurgen, O. Hellmuth, B. Froba, T. Kastner, and M. Cremer. Content-based Identification of Audio Material using MPEG-7 Low Level Description. In *Proc. Int. Symposium on Music Information Retrievla (ISMIR)*, 2001.

[6] D. Anastassiou. Genomic Signal Processing. *IEEE Signal Processing Magazine*, 18(4):8–21, July 2001.

[7] 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.

[8] J.-J. Aucouturier and M. Sandler. Segmentation of Musical Signals Using Hidden Markov Models. In *Proc. 110th Audio Engineering Society Convention*, Amsterdam, The Netherlands, May 2001. Audio Engineering Society AES.

[9] D. H. Ballard and C. M. Brown. *Computer Vision*. Prentice Hall, 1982.

[10] M. A. Bartsch and G. H. Wakefield. To Catch a Chorus: Using Chroma-Based Representation for Audio Thumbnailing. In *Proc. Int. Workshop on applications of Signal Processing to Audio and Acoustics*, pages 15–19, Mohonk, NY, 2001. IEEE.

[11] S. Belongie, C. Carson, H. Greenspan, and J. Malik. Blobworld: A system for region-based image indexing and retrieval. In *Proc. 6th Int. Conf. on Computer Vision*, Jan. 1998.

[12] A. L. Berenzweig and D. P. Ellis. Locating singing voice segments within musical signals. In *Proc. Int. Workshop on Applications of Signal Processing to Audio and Acoustics WASPAA*, pages 119–123, Mohonk, NY, 2001. IEEE.

[13] J. Biles. GenJam: A Genetic Algorithms for Generating Jazz Solos. In *Proc. Int. Computer Music Conf. (ICMC)*, pages 131–137, Aarhus, Denmark, Sept. 1994.

[14] G. Boccignone, M. DeSanto, and G. Percannella. Joint Audio-Video Processing of MPEG Encoded Sequences. In *Proc. Int. Con.f On Multimedia Computing and Systems (ICMCS)*, pages 225–229. IEEE, 1999.

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

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

[17] J. Brown. Computer identification of musical instruments. *Journal of the Acoustical Society of America*, 105(3):1933–1941, 1999.

[18] C. Chafe, B. Mont-Reynaud, and L. Rush. Toward an Intelligent Editor of Digital Audio: Recognition of Musical Constructs. *Computer Music Journal*, 6(1):30–41, 1982.

[19] P. Cook. Physically inspired sonic modeling (PHISM): synthesis of percussive sounds. *Computer Music Journal*, 21(3), Aug. 1997.

[20] P. Cook. Toward physically-informed parametric synthesis of sound effects. In *Proc. IEEE Workshop on applications of Signal Processing to Audio and Acoustics, WASPAA*, New Paltz, NY, 1999. Invited Keynote Address.

[21] P. Cook, editor. *Music, Cognition, and Computerised Sound.* MIT Press, 2001.

[22] P. Cook, G. Essl, and D. Trueman. N >> 2: Multi-speaker Display Systems for Virtual Reality and Spatial Audio Projection. In *Proc. Int. Conf. on Auditory Display (ICAD)*, Glasgow, Scotland, 1998.

[23] P. Cook and G. Scavone. The Synthesis Toolkit (STK), version 2.1. In *Proc. Int. Computer Music Conf. ICMC*, Beijing, China, Oct. 1999. ICMA.

[24] R. Dannenberg. An on-line algorithm for real-time accompaniment. In *Proc. Int. Computer Music Conf.*, pages 187–191, Paris, France, 1984.

[25] I. Daubechies. Orthonormal bases of compactly supported wavelets. *Communications on Pure and Applied Math*, 41:909–996, 1988.

[26] S. Davis and P. Mermelstein. Experiments in syllable-based recognition of continuous speech. *IEEE Transcactions on Acoustics, Speech and Signal Processing*, 28:357–366, Aug. 1980.

[27] H. Deshpande, R. Singh, and U. Nam. Classification of Musical Signals in the Visual Domain. In *Proc. COST G-G Conf. on Digital Audio Effects (DAFX)*, Limerick, Ireland, Dec. 2001.

[28] S. Dixon. A Lightweight Multi-agent Musical Beat Tracking System. In *Proc. Pacific Rim Int. Conf. on Artificial Intelligence*, pages 778–788, 2000.

[29] S. Dixon. An Interactive Beat Tracking and Visualization System. In *Proc. Int. Computer Music Conf. (ICMC)*, pages 215–218, Habana, Cuba, 2002. ICMA.

[30] R. Duda, P. Hart, and D. Stork. *Pattern classification*. John Wiley & Sons, New York, 2000.

[31] D. Ellis. *Prediction-driven computational auditory scene analysis*. PhD thesis, MIT Media Lab, 1996.

[32] A. Eronen and A. Klapuri. Musical Instrument Recognition using Cepstral Features and Temporal Features. In *Int. Conf. on Acoustics, Speech and Signal Processing ICASSP*, Istanbul Turkey, 2000. IEEE.

[33] G. Essl and P. Cook. Measurements and efficient simulations of bowed bars. *Journal of Acoustical Society of America (JASA)*, 108(1):379–388, 2000.

[34] U. Fayyad, G. G. Grinstein, and A. Wierse, editors. *Information Visualization in Data Mining and Knowledge Discovery*. Morgan Kaufmann, 2002.

[35] M. Fernstorm and C. McNamara. After Direct Manipulation - Direct Sonification. In *Proc. Int. Conf. on Auditory Display, ICAD*, Glasgow, Scotland, 1998.

[36] M. Fernstrom and E. Brazil. Sonic Browsing: an auditory tool for multimedia asset management. In *Proc. Int. Conf. on Auditory Display (ICAD)*, Espoo, Finland, July 2001.

[37] M. Flickner and et al. Query by image and video content: the QBIC system. *IEEE Computer*, 28(9):23–32, Sept. 1995.

[38] J. Foote. Content-based retrieval of music and audio. In *Multimedia Storage and Archiving Systems II*, pages 138–147, 1997.

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

[40] J. Foote. Visualizing music and audio using self-similarity. In *ACM Multimedia*, 1999.

[41] J. Foote. Arthur: Retrieving orchestral music by long-term structure. Read at the First International Symposium on Music Information Retrieval, 2000.

[42] J. Foote. Automatic Audio Segmentation using a Measure of Audio Novelty. In *Proc. Int. Conf. on Multimedia and Expo*, volume 1, pages 452–455. IEEE, 2000.

[43] J. Foote and S. Uchihashi. The Beat Spectrum:a new approach to rhythmic analysis. In *Int. Conf. on Multimedia & Expo*. IEEE, 2001.

[44] I. Fujinaga. Machine recognition of timbre using steady-state tone of acoustic instruments. In *Proc. Int. Computer Music Conf. ICMC*, pages 207–210, Ann Arbor, Michigan, 1998. ICMA.

[45] I. Fujinaga. Realtime recognition of orchestral instruments. In *Int. Computer Music Conf. ICMC*, 141-143. ICMA, 2000.

[46] B. Garton. Virtual Performance Modelling. In *Proc. Int. Computer Music Conf. (ICMC)*, pages 219–222, San Jose, California, Oct. 1992.

[47] A. Ghias, J. Logan, D. Chamberlin, and B. Smith. Query by Humming: Musical Information Retrieval in an Audio Database. *ACM Multimedia*, pages 213–236, 1995.

[48] M. Goto and Y. Muraoka. Real-time rhythm tracking for drumless audio signals - chord change detection for musical decisions. In *Proc. Int. Joint. Conf. in Artificial Intelligence: Workshop on Computational Auditory Scene Analysis*, 1997.

[49] M. Goto and Y. Muraoka. Music Understanding at the Beat Level: Real-time Beat Tracking of Audio Signals. In D. Rosenthal and H. Okuno, editors, *Computational Auditory Scene Analysis*, pages 157–176. Lawrence Erlbaum Associates, 1998.

[50] J. M. Gray. *An Exploration of Musical Timbre*. PhD thesis, Dept. of Psychology, Stanford Univ., 1975.

[51] W. Grosky, R. Jain, and R. Mehrotra, editors. *The handbook of Multimedia Information Management*. Prentice Hall, 1997.

[52] T. Hastie and W. Stuetzle. Principal curves. *Journal of the American Statistical Association*, 84(406):502–516, 1989.

[53] 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, 1997. http://www.cs.cmu.edu/afs/cs/user/alex/www/.

[54] T. Hermann, P. Meinicke, and H. Ritter. Principal Curve Sonification. In *International Conference on Auditory Display*, ICAD, 2000.

[55] R. E. Johnson. Components, Frameworks, Patterns. In *Proc. ACM SIGSOFT Symposium on Software Reusability*, pages 10–17, 1997.

[56] L. Jolliffe. *Principal Component Analysis*. Springer-Verlag, New York, 1986.

[57] J. M. Jose, J. Furner, and D. J. Harper. Spatial querying for image retrieval: a user-oriented evaluation. In *Proc. SIGIR Conf. on research and development in Information Retrieval*, Melbourne, Australia, 1998. ACM.

[58] I. JTC1/SC29. *Information Technology-Coding of Moving Pictures and Associated Audio for Digital Storage Media at up to about 1.5 Mbit/s-IS 11172 (Part 3, Audio)*. 1992.

[59] I. JTC1/SC29. *Information Technology-Generic Coding of Moving Pictures and Associated Audio Information-IS 13818 (Part 3, Audio)*. 1994.

[60] H. Kang and B. Shneiderman. Visualization Methods for Personal Photo Collections: Browsing and Searching in the PhotoFinder. In *Proc. Int. Con.f on Multimedia and Expo*, New York, 2000. IEEE.

[61] N. Kashino and T. Kinoshita. Application of Bayesian probability network to music scene analysis. In *Proc. Int. Joint Conf. On Artificial Intelligence, CASA Workshop*, 1995.

[62] T. Kemp, M. Schmidt, M. Westphal, and A. Waibel. Strategies for automatic segmentation of audio data. In *Proc. Int. Conf. on Acoustics Speech and Signal Processing (ICASSP)*, volume 3, pages 1423–1426. IEEE, 2000.

[63] D. Kimber and L. Wilcox. Acoustic segmentation for audio browsers. In *Interface Conference*, pages Sydney, Australia, July 1996.

[64] G. E. Krasner and S. T. Pope. A cookbook for using the model-view-controller user interface paradigm in Smalltalk-80. *Journal of Object-Oriented Programming*, 1(3):26–49, Aug. 1988.

[65] R. Kronland-Martinet, J. Morlet, and A. Grossman. Analysis of sound patterns through wavelet transforms. *Int. Journal of Pattern Recognition and Artificial Intelligence*, 1(2):237–301, 1987.

[66] J. Laroche. Estimating Tempo, Swing and Beat Locations in Audio Recordings. In *Proc. Int. Workshop on applications of Signal Processing to Audio and Acoustics WASPAA*, pages 135–139, Mohonk, NY, 2001. IEEE.

[67] F. Lerdahl and R. Jackendoff. *A Generative Theory of Tonal Music*. MIT Press, 1983.

[68] G. Li and A. Khokar. Content-based indexing and retrieval of audio data using wavelets. In *Int. Conf. on Multimedia and Expo (II)*, pages 885–888. IEEE, 2000.

[69] K. Li and et. al. Building and using a scalable display wall system. *IEEE Computer Graphics and Applications*, 21(3), July 2000.

[70] S. Li. Content-based classification and retrieval of audio using the nearest feature line method. *IEEE Transactions on Speech and Audio Processing*, 8(5):619–625, Sept. 2000.

[71] J. List, A. van Ballegooij, and A. de Vries. Known-Item Retrieval on Broadcast TV. Technical Report INS-R0104, CWI, Apr. 2001.

[72] B. Logan. Mel Frequency Cepstral Coefficients for Music Modeling. In *Proc. Int. Symposium on Music Information Retrieval (ISMIR)*, 2000.

[73] B. Logan. Music summarization using key phrases. In *Proc. Int. Conf. on Acoustics, Speech and Signal Processing ICASSP*. IEEE, 2000.

[74] L. Lu, J. Hao, and Z. HongJiang. A robust audio classification and segmentation method. In *Proc. ACM Multimedia*, Ottawa, Canada, 2001.

[75] P. C. Mahalanobis. On the generalized distance in statistics. In *Proc. National Inst. of Science India*, volume 12, pages 49–55, 1936.

[76] J. Makhoul. Linear prediction: A tutorial overview. *Proceedings of the IEEE*, 63:561–580, Apr. 1975.

[77] S. G. Mallat. *A Wavelet Tour of Signal Processing*. Academic Press, 1999.

[78] G. Marchionini. *Information Seeking in Electronic Environments*. The Press Syndicate of the University of Cambridge, 1995.

[79] K. Martin. A Blackboard System for Automatic Transcription of Simple Polyphonic Music. Technical Report 399, MIT Media Lab, 1996.

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

[81] K. Martin. *Sound-Source Recognition: A Theory and Computational Model*. PhD thesis, MIT Media Lab, 1999.

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

[83] N. Masako and K. Watanabe. Interactive Music Composer based on Neural Networks. In *Proc. Int. Computer Music Conf. (ICMC)*, San Jose, California, 1992.

[84] T. K. Moon. The Expectation-Maximization Algorithm. *IEEE Signal Processing Magazine*, 13(6):47–60, Nov. 1996.

[85] F. Moore. *Elements of Computer Music*. Prentice Hall, 1990.

[86] J. Moorer. *On the Segmentation and Analysis of Continuous Musical Sound by Digital Computer*. PhD thesis, Dept. of Music, Stanford University, 1975.

[87] J. Moorer. The Lucasfilm Audio Signal Processor. *Computer Music Journal*, 6(3):30–41, 1982. (also in ICASSP 82).

[88] P. Noll. MPEG digital audio coding. *IEEE Signal Processing Magazine*, pages 59–81, September 1997.

[89] A. Oppenheim and R. Schafer. *Discrete-Time Signal Processing*. Prentice Hall, Edgewood Cliffs, NJ, 1989.

[90] L. Ottaviani and D. Rocchesso. Separation of Speech Signal from Complex Auditory Scenes. In *Proc. COST G-6 Conf. on Digital Audio Effects (DAFX)*, Limerick, Ireland, Dec. 2001.

[91] A. Pentland, R. Picard, and S. Sclaroff. Photobook: Tools for Content-Based Manipulation of Image Databases. *IEEE Multimedia*, pages 73–75, July 1994.

[92] D. Perrot and R. Gjerdigen. Scanning the dial: An exploration of factors in identification of musical style. In *Proc. Society for Music Perception and Cognition*, page 88, 1999. (abstract).

[93] S. Pfeiffer. Pause concepts for audio segmentation at different semantic levels. In *Proc. ACM Multimedia*, Ottawa, Canada, 2001.

[94] J. Pierce. Consonance and Scales. In P. Cook, editor, *Music Cognition and Computerized Sound*, pages 167–185. MIT Press, 1999.

[95] D. Pye. Content-based methods for the management of digital music. In *Proc. Int. Conf on Acoustics, Speech and Signal processing ICASSP*. IEEE, 2000.

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

[97] L. Rabiner and B. Gold. *Theory and Application of Digital Signal Processing*. Prentice Hall, 1975.

[98] L. Rabiner and B. H. Juang. *Fundamentals of Speech Recognition.* Prentice-Hall, 1993.

[99] C. Roads. *Computer Music Tutorial*. MIT Press, 1996.

[100] K. Rodden, W. Basalaj, D. Sinclair, and K. Wood. Does Organization by Similarity Assist Image Browsing ? In *Proc. SIG-CHI on Human Factors in computing systems*, Seattle WA, USA, 2001. ACM.

[101] D. F. Rosental and H. G. Okuno, editors. *Computational Auditory Scene Analysis*. Lawrence Erlbaum, 1998.

[102] S. Rossignol, X. Rodet, et al. Features extraction and temporal segmentation of acoustic signals. In *Proc. Int. Computer Music Conf. ICMC*, pages 199–202. ICMA, 1998.

[103] J. Saunders. Real time discrimination of broadcast speech/music. In *Proc. Int. Conf. on Acoustics, Speech and Signal Processing ICASSP*, pages 993–996. IEEE, 1996.

[104] G. Scavone, S. Lakatos, P. Cook, and C. Harbke. Perceptual spaces for sound effects obtained with an interactive similarity rating program. In *Proc. Int. Symposium on Musical Acoustics*, Perugia, Italy, Sept. 2001.

[105] R. Schalkoff. *Pattern Recognition. Statistical, Structural and Neural Approaches*. John Wiley & Sons, 1992.

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

[107] E. Scheirer. The MPEG-4 structured audio standard. In *Proc. Int. Conf. on Acoustics, Speech and Signal Processing ICASSP*. IEEE, 1998.

[108] E. Scheirer. Tempo and beat analysis of acoustic musical signals. *Journal of the .Acoustical Society of America*, 103(1):588,601, Jan. 1998.

[109] E. Scheirer. *Music-Listening Systems*. PhD thesis, MIT, 2000.

[110] E. Scheirer and M. Slaney. Construction and evaluation of a robust multifeature speech/music discriminator. In *Proc. Int. Conf. on Acoustics, Speech and Signal Processing ICASSP*, pages 1331–1334. IEEE, 1997.

[111] D. Schwarz. A system for data-driven concatenative sound synthesis. In *Proc. Cost-G6 Conf. on Digital Audio Effects (DAFX)*, Verona, Italy, Dec. 2000.

[112] J. Seppänen. Quantum Grid Analysis of Musical Signals. In *Proc. Int. Workshop on applications of Signal Processing to Audio and Acoustics WASPAA*, pages 131–135, Mohonk, NY, 2001. IEEE.

[113] R. N. Shepard. Circularity in Judgements of Relative Pitch. *Journal of the Acoustical Society of America*, 35:2346–2353, 1964.

[114] B. Shneiderman. *Designing the User Interface: Strategies for Effective Human-Computer Interaction*. Addison-Wesley, 3rd ed. edition, 1998.

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

[116] M. Slaney and R. Lyon. A perceptual pitch detector. In *Proc. Int. Conf. on Acoustics, Speech and Signal Processing ICASSP*, pages 357–360, Albuquerque, NM, 1990. IEEE.

[117] 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.

[118] P. Smaragdis. *Redundancy Reduction for Computational Audition, a Unifying Approach*. PhD thesis, MIT Media Lab, 2001.

[119] L. Smith. *A Multiresolution Time-Frequency Analysis And Interpretation Of Musical Rhythm*. PhD thesis, University of Western Australia, July 1999.

[120] R. Spence. *Information Visualization*. Addison Wesley ACM Press, 2001.

[121] K. Steiglitz. *A digital signal processing primer*. Addison Wesley, 1996.

[122] A. Subramanya, S.R. annd Youssef. Wavelet-based indexing of audio data in audio/multimedia databases. In *Proc. Int. Workshop on Multimedia Database Management IW-MMDBMS*, pages 46–53, 1998.

[123] T. Tolonen and M. Karjalainen. A Computationally Efficient Multipitch Analysis Model. *IEEE Trans. on Speech and Audio Processing*, 8(6):708–716, Nov. 2000.

[124] G. Tzanetakis and P. Cook. A Framework for Audio Analysis based on Classification and Temporal Segmentation. In *Proc. Euromicro 99, Workshop on Music Technology and Audio Processing*, 1999.

[125] G. Tzanetakis and P. Cook. Multifeature audio segmentation for browsing and annotation. In *Proc. Workshop on applications of signal processing to audio and acoustics WASPAA*, New Paltz, NY, 1999. IEEE.

[126] G. Tzanetakis and P. Cook. 3D Graphics Tools for Sound Collections. In *Proc. COST G6 Conf. on Digital Audio Effects (DAFX)*, Verona, Italy, Dec. 2000.

[127] G. Tzanetakis and P. Cook. Audio Information Retrieval (AIR) Tools. In *Proc. Int. Symposium on Music Information Retrieval (ISMIR)*, 2000.

[128] G. Tzanetakis and P. Cook. Experiments in computer-assisted annotation of audio. In *Proc. Int. Con. on Auditory Display*, ICAD, 2000.

[129] G. Tzanetakis and P. Cook. Marsyas: A framework for audio analysis. *Organised Sound*, 4(3), 2000.

[130] G. Tzanetakis and P. Cook. Sound analysis using MPEG compressed audio. In *Proc. Int. Conf. on Acoustics, Speech and Signal Processing ICASSP*, Istanbul, 2000. IEEE.

[131] G. Tzanetakis and P. Cook. Marsyas3D: a prototype audio browser-editor using a large scale immersive visual and audio display. In *Proc. Int. Conf. on Auditoy Display (ICAD)*, Espoo, Finland, Aug. 2001.

[132] G. Tzanetakis and P. Cook. *Audio Information Retrieval using Marsyas*. Kluewe Academic Publishers, 2002. (to be published).

[133] G. Tzanetakis and P. Cook. Musical Genre Classification of Audio Signals. *IEEE Transactions on Speech and Audio Processing*, 2002. (accepted for publication).

[134] G. Tzanetakis, G. Essl, and P. Cook. Audio Analysis using the Discrete Wavelet Transform. In *Proc. Conf. in Acoustics and Music Theory Applications*. WSES, Sept. 2001.

[135] G. Tzanetakis, G. Essl, and P. Cook. Automatic Musical Genre Classification of Audio Signals. In *Proc. Int. Symposium on Music Information Retrieval (ISMIR)*, Oct. 2001.

[136] G. Tzanetakis and L. Julia. Multimedia Structuring using Trees. In *Proc. In Proc. RIAO 2000 ”Content based multimedia information access”*, Paris, France, Apr. 2001.

[137] K. van Rijsbergen. *Information retrieval*. Butterworths, London, 2nd edition, 1979.

[138] R. Vertegaal and E. Bonis. ISEE: An Intuitive Sound Editing Environment. *Computer Music Journal*, 18(2):21–29, 1994.

[139] R. Vertegaal and B. Eaglestone. Looking for Sound ? Selling Perceptual Space in Hierarchically Nested Boxes. In *Summary CHI 98 Conf. on Human Factors in Computing Systems*. ACM, 1998.

[140] S. Wake and T. Asahi. Sound Retrieval with Intuitive Verbal Expressions. In *Proc. Int. Conf. on Auditory Display (ICAD)*, Glasgow, Scotland, 1997.

[141] Y. Wang and V. Miikka. A compressed domain beat detector using MP3 audio bitstreams. In *Proc. ACM Multimedia*, Ottawa, Canada, 2001.

[142] B. Whitman, G. Flake, and S. Lawrence. Artist Detection in Music with Minnowmatch. In *Proc. Workshop on Neural Networks for Signal Processing*, pages 559–568, Falmouth, Massachusetts, Sept. 2001. IEEE.

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

[144] C. Yang. MACS: Music Audio Characteristic Sequence Indexing for Similarity Retrieval. In *Proc. Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, New Paltz, New York, 2001. IEEE.

[145] C. Yang. Music Database Retrieval based on Spectral Similarity. In *Proc. Int. Symposium on Music Information Retrieval (Poster) (ISMIR)*, Bloomington, Indiana, 2001.

[146] B. Yeo and B. Liu. Rapid Scene Analysis on Compressed Videos. *IEEE Trans. Circuits System. Video Technology*, 5(6), 1995.

[147] T. Zhang and J. Kuo. Audio Content Analysis for online Audiovisual Data Segmentation and Classification. *Transactions on Speech and Audio Processing*, 9(4):441–457, May 2001.

[148] A. Zils and F. Pachet. Musical Mosaicing. In *Proc. Cost-G6 Conf. on Digital Audio Effects (DAFX)*, Limerick, Ireland, Dec. 2001.