UNIVERSITY OF SOUTHAMPTON

Faculty of Physical Sciences and Engineering

Electronics and Computer Science

Content Based Retrieval and Classification of Music Using Polyphonic Timbre Similarity

by

Franz Asunta de Leon

Thesis for the degree of Doctor of Philosophy

May 2014

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF PHYSICAL SCIENCES AND ENGINEERING

Electronics and Computer Science

Thesis for the degree of Doctor of Philosophy

CONTENT BASED RETRIEVAL AND CLASSIFICATION OF MUSIC USING POLYPHONIC TIMBRE SIMILARITY

Franz Asunta de Leon

Digital technology and the Internet have changed the music industry's landscape. Music has become more accessible allowing consumers to store and share thousands of items in their computer's hard disk, portable media player, mobile phone and other devices. Recent developments allow consumers to store digital music on the Internet through cloud storage. Given the large music collections available, there is a need for new applications for browsing, organising, discovering, and generating playlists for users.

In previous years, searching for music has been similar to a textual information search. However, this limits music discovery as it usually requires specific information that may be unknown to the user. This thesis investigates three of the core components of content-based music retrieval: audio features, similarity functions and indexing methods. In the content-based paradigm, audio files are analyzed using their waveform and are represented by high-dimensional features. This study focuses on polyphonic timbre similarity. Polyphonic timbre is the characteristic that allows listeners to differentiate between two music signals or complex instrumental textures with the same perceived pitch and loudness. The different attributes of timbre are examined and suitable features that can be used for music retrieval using timbre similarity are investigated. Evaluations are performed to compare the performance of these features. To improve the overall performance and reduce the undesirable effects of operating in high-dimensionality space, methods on how feature spaces can be combined are also explored.

A full linear scan of the feature space is impractical for large music collections. Hence, the filter-and-refine method is adopted to expedite the retrieval process. The objective is to filter a dataset by quickly returning a set of candidate songs then refining the results using an exact similarity measure. Some novel modifications of the filtering step are made to ensure that the level of performance is maintained. The application of our timbre similarity systems are extended to automatic audio classification. In this paradigm, an unlabeled track is tagged with the label of the nearest track. Finally, the performance of our similarity estimator and audio classifier are validated in the annual Music Information Retrieval Evaluation eXchange (MIREX). The MIREX results show that our techniques are state-of-the-art methods.

Contents

List of tables

List of figures

DECLARATION OF AUTHORSHIP

I, Franz Asunta de Leon

declare that the thesis entitled

Content Based Retrieval and Classification of Music Using Polyphonic Timbre Similarity

and the work presented in the thesis are both my own, and have been generated by me as the result of my own original research. I confirm that:

- this work was done wholly or mainly while in candidature for a research degree at this University;
- where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated;
- where I have consulted the published work of others, this is always clearly attributed;
- where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work;
- I have acknowledged all main sources of help;
- where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself;
- parts of this work have been published as:

    1. F. de Leon and K. Martinez. A music genre classifier combining timbre, rhythm, and tempo. In *Proc. of 2012 IEEE Region 10 Conference*, Cebu, Philippines, November 2012.
    2. F. de Leon and K. Martinez. Towards efficient music genre classification using FastMap. In *Proc. of 19th International Conference on Audio Effects*, York, United Kingdom, September 2012.
    3. F. de Leon and K. Martinez. Enhancing timbre model using MFCC and its time derivatives for music similarity estimation. In *Proc. of the 20th European Signal Processing Conference*, pp. 2005-2009, Bucharest, Romania, August 2012.

Signed:……..

Date:……

Acknowledgements

I would like to thank the following faculty members of the lab: Kirk Martinez for his support and supervision, Dave de Roure for giving me the chance to work in the field of Music Information Retrieval, and Paul Lewis for serving as my internal examiner for my 9-month report and transfer viva. I would also like to extend my gratitude to Michael Kraft for helping during my admission process in the University.

I would like to thank Mark Sandler and Jonathan Hare, my external and internal examiners for the final viva, for giving constructive comments to improve my work.

I would like to thank Rowena Guevara and Joel Marciano of the University of the Philippines Electrical and Electronics Engineering Institute, and the Engineering Research and Development for Technology of the Department of Science and Technology for supporting my doctoral studies.

I would like to thank the MIREX team from the University of Illinois at Urbana-Champaign, in particular J. Stephen Downie and Yun Hao, for their patience during the evaluations.

Finally, I would like to thank my family and friends, especially my wife Tess, for their unwavering love and support. I am also dedicating this work to our first child for serving as one of my inspirations.

Definitions and Abbreviations

x	Time domain audio vector
$\overline{x}$	Mean value of the audio vector x
w	Window function
x_{seg}	Segmented audio vector
x'_{seg}	Segmented audio vector after applying window function
X	Discrete Fourier transform of f
c_t	Cepstral coefficient at time frame t
Θ	Time window in terms of frames
d_t	Delta coefficients at time t
dd_t	Delta-delta coefficients at time t
$peak_k$	Spectral peak of the k-th sub-band
$valley_k$	Spectral valley of the k-th sub-band
SC_k	Spectral contrast of the k-th sub-band
$D_E(p,q)$	Euclidean distance between vectors p and q
M_t	Spectral magnitude at frame t
μ_1	Spectral centroid
μ_2	Spectral spread
μ_3	Spectral skewness
μ_4	Spectral kurtosis
SF	Spectral flatness
$SFlx$	Spectral flux
R	Spectral roll-off
SB	Spectral brightness
SE	Spectral entropy
μ	Mean of vectors
Σ	Covariance matrix
$N(\mu,\Sigma)$	Gaussian probability density function with mean μ and covariance matrix Σ
$Tr()$	Matrix trace
$\lvert\Sigma\rvert$	Determinant of matrix Σ
$KL(p \mid q)$	Kullback-Leibler divergence between distributions p and q
$SKL(p \mid q)$	Symmetrised Kullback-Leibler divergence between distributions p and q
$\delta(p \mid q)$	Metric approximation of symmetrised Kullback-Leibler divergence between distributions p and q
P_n	Precision at n items

$d_{x,y}$	Distance matrix value at row x and column y
$\mu_{x,y}$	Mean of values at row x and column y of a distance matrix
$\sigma_{x,y}$	Standard deviation of values at row x and column y of a distance matrix
$d'_{x,y}$	Gaussian normalised distance matrix value at row x and column y
$N_k(i)$	Number of times song i appears among the k nearest neighbours
ω	Scalar weight

ADC	Analogue-to-Digital
AMS	Audio Music Similarity
ASA	American Standards Association
ASR	Automatic Speech Recognition
DAG	Directed Acyclic Graph
DBN	Deep Belief Network
DCT	Discrete Cosine Transform
DFT	Discrete Fourier Transform
EM	Expectation-Maximization
EMD	Earth Mover's Distance
FFT	Fast Fourier Transform
FP	Fluctuation Pattern
GMM	Gaussian Mixture Model
HMM	Hidden Markov Model
IFPI	International Federation of the Phonographic Industry
IMIRSEL	International Music Information Retrieval System Evaluation Laboratory
ISMIR	International Society for Music Information Retrieval
KL	Kullback-Leibler
k-NN	k-Nearest Neighbours
LPC	Linear Predictive Coding
MDS	Multidimensional Scaling
MFCC	Mel-frequency Cepstral Coefficients
MIDI	Musical Instrument Digital Interface
MIR	Music Information Retrieval
MIREX	Music Information Retrieval Evaluation eXchange
MP	Mutual Proximity
NICDM	Non-Iterative Contextual Dissimilarity Measure

PCA	Principal Component Analysis
PCM	Pulse-Code Modulation
PDF	Probability Density Function
RBF	Radial Basis Function
RBM	Restricted Boltzmann Machine
SKL	Symmetrised Kullback-Leibler
STFT	Short-time Fourier Transform
SVD	Singular Value Decomposition
SVM	Support Vector Machine
VQ	Vector Quantization

Chapter 1: Introduction

Before the age of digital music and the Internet, searching for music was a more personal experience for consumers. People visit their favourite record shop to search for new music, and even ask for recommendations from in-house experts based on their personal preference. The records are usually arranged by genre, and albums from the same artists are grouped together for easier browsing. Sometimes it is also possible to preview some tracks before finally making a purchase. Now, it is more convenient for people to discover music online and buy music for home delivery or digital downloads.

A common method for searching for music, whether from a digital library or online, is by keyword search. People can just enter the artist, track title or album name on a search engine and a list of nearest matching tracks are returned. However, this assumes that the user knows specific keywords and that to some extent such tracks exist. It is also possible to perform a broader search by providing general keywords, such as genre or mood. This assumes that all the tracks are accurately tagged to increase the chances of returning the desired music. There are other possibilities that exploit the nature of music itself. This may involve searching using the following queries: audio clips, pitch contour, singing/humming, tapping, etc. In general, it is expected that the music search and retrieval system can handle the complexities of music but with the ease of use of textual search engines.

1.1 Motivation

Digital technology and the Internet have changed the music industry landscape. Music has become more accessible, allowing consumers to store and share thousands of items in their computer's hard disk, portable media player, mobile phone and other devices. Recent developments also allow storage of digital music in the Internet through cloud

storage. This led to music distributors to establishing online channels such as the Apple iTunes Store[1], Amazon MP3[2] and Spotify[3].

In 2013, the International Federation of the Phonographic Industry reported that global recorded music industry revenues decreased by an estimated 8.5% to US$15.1 billion (IFPI 2013). Digital revenues continued its growth at 5% to US$5.9 billion in 2013; this translates to 39% of global industry revenues. The number of people using paid subscription services jumped by 40% to 28 million. In 2011, major digital retailers were present in 23 countries. After three years, they are now present in more than 150 countries. These data reflect the growing optimism for the recovery of the music industry, thanks to the adoption of new technologies. In return, music is pushing the technology to develop devices, drive online search and social networking, and the demand for fast broadband connections. Music has become a major element of the digital ecosystem.

Given the large music datasets available, there is a need for new applications for browsing, organising, discovering, as well as generating playlists for users. Let us consider some of the online music sites that feature a recommendation system. Pandora Radio[4] is a music recommendation service based on the Music Genome Project. The Music Genome Project involves experts who assign up to 400 different musical attributes to every song. Pandora recommends a playlist based on user feedback and considers the attributes when selecting the next song to be played. This technique of using user feedback is called collaborative filtering. In collaborative filtering, music files are recommended to a user based on the preference of other, similar users (Cohen 2000). Although the results of this method are often satisfactory, the approach doesn't scale very well as it is time consuming to assign attributes to each song.

---

[1] http://www.apple.com/itunes

[2] http://www.amazon.co.uk/MP3

[3] http://www.spotify.com

[4] http://www.pandora.com

Another application for music recommendation but this time, purely based on algorithms, is Mufin[5]. Mufin's software extracts properties of a song and makes recommendation based entirely on the music similarity between songs. It can also provide 3D visualization of the music collection by displaying each track as a circle. Each axis in the 3D space is defined as follows: sad to happy (left to right), synthetic to acoustic (bottom to top), and from calm to aggressive (front to back). A technology review of Mufin Player reports that the software can make unexpected recommendations based on the sound of the music rather than the artificial genre categories that artists and labels apply (Rosoff 2009).

The previous examples illustrate how services adapt to the changing music landscape. There are other challenges that present new opportunities for research using large music collections for searching and retrieval of music-related data. This research field is collectively called Music Information Retrieval (MIR). MIR is a multidisciplinary field that includes acoustics, psychoacoustics, signal processing, computer science, musicology, library science, informatics, and machine learning, etc. (Downie 2008). Its main goal is to provide a level of access to the world's vast music collection on a level at par with, or exceeding that of text-based search engines. This report focuses on content-based methods for sound similarity estimation and classification. This involves extracting useful information or features from audio signals for these two tasks. This study investigates issues and limitations involved with the content-based approach.

1.2 Objectives

The main objective of this research is to investigate how content-based methods can be used to perform polyphonic timbre similarity estimation. Polyphonic timbre refers to the global characteristic that allows listeners to differentiate between two music signals or complex instrumental textures with the same perceived pitch and loudness (Aucouturier et al. 2005). Timbre similarity estimation is important for playlist recommendation,

[5] http://www.mufin.com

indexing and retrieval applications, etc. The secondary objective of this work is to apply suitable methods that will allow content-based audio retrieval to be used with large collections. This goal leads us to research the properties of three core components of content-based audio retrieval. These are: 1) the audio features used to extract timbre from a waveform, 2) the functions used to quantify the similarity between features, and 3) the indexing methods used to expedite a search. The two key performance measures addressed are retrieval precision and speed of search. Retrieval systems that can achieve high precisions at realistic timescales are required for commercial applications.

Both the speed and precision of the content-based retrieval method can be compromised by the high-dimensionality of the features that represent timbre. The effects of high dimensionality can reduce the performance of the nearest neighbour search and the effectiveness of indexing techniques. However, the performance of signal processing methods is known to approach a 'glass ceiling', i.e. a performance limit that cannot be overcome by system variations. This work is done keeping in mind that it serves to complement other approaches that require more time and resources, such as manual annotation of music tracks. It recognizes that audio similarity is very much dependent on user cultural background or preferences, and audio classification is best done by musical experts.

1.3 Contributions

This thesis brings a number of clear contributions to the field of music information retrieval. These contributions are briefly enumerated below.

- A comparison of Mel-frequency cepstral coefficients, spectral contrast, sub-band flux, and spectral distribution descriptors for timbre similarity estimation
- Development of a music similarity system using three feature spaces based on attributes of timbre
- Improvement of the spectral envelope feature space by appending spectral distribution descriptors
- Demonstration of a 3-d polyphonic timbre space where each dimension corresponds to an attribute of timbre
- Development of a linear combination technique for combining the feature spaces

- Development of an efficient retrieval system by adopting the filter-and-refine method, including two novel variations of the filtering step
- Demonstration of a technique for audio classification by propagating the nearest neighbor's label

This research has led to three refereed conference publications on varying subjects, and two poster presentations for our submission to the annual Music Information Retrieval Evaluation eXchange (MIREX). We proposed the idea of enhancing the timbre model by considering MFCCs and its time derivatives to improve genre classification accuracies (de Leon and Martinez 2012a). This method is later used between MFCCs and other timbre feature spaces to improve the performance of our timbre similarity system. We also proposed an efficient music genre classifier that adopts the filter-and-refine method (de Leon and Martinez 2012b). The proposed system uses a combination of FastMap algorithm and Kullback-Leibler divergence to return the nearest neighbour whose label is used to classify an unlabeled song. We also described a music similarity estimator based on timbre, rhythm, and tempo (de Leon and Martinez 2012c). The similarity estimator was submitted to the 2011 MIREX audio music similarity task for validation. Based on the results[6], the neighbourhood clustering accuracy in the top 5 retrieved items according to artist and genre similarity is 0.35 and 0.68, respectively. In comparison, the best submission[7] obtained 0.40 and 0.7, respectively. Recognising the potential for improvements in our system, we redirected our focus on improving the timbre models.

1.4 Document Structure

This report describes the work of the author to achieve the aims and objectives outlined earlier in this chapter. Chapter 2 gives a background on the methods for searching music and describe the existing literature towards content-based audio retrieval. Chapters 3 through 5 describe the actual research done by the author, and Chapter 6 presents the

[6] http://music-ir.org/mirex/results/2011/ams/statistics/DM2.results/report.txt
[7] http://music-ir.org/mirex/results/2011/ams/statistics/SSPK2.results/report.txt

conclusions of this research, including its limitations, as well as suggesting directions for further studies. The structure and content of each chapter are discussed in detail below.

Chapter 2 – The role of timbre in music search. This chapter first discusses the role of timbre in music search. There are very few content-based applications for music retrieval. The main problem is the semantic gap between a user requirement and the content-based representation. To counter this, several approaches to music retrieval are discussed to highlight their advantages and disadvantages. From this we see that there is a genuine need for content-based approaches. A general content-based query system is then described followed by a discussion of each component.

As this research focuses on modelling timbre, the complex nature of timbre is described. We enumerate the different definitions of timbre, from the standard definition to a definition that can be understood by ordinary people. From psychoacoustic studies, they have identified several attributes of timbre. These attributes will be the foundation of the features that are extracted in this work. We discuss the feature extractions methods that represent the different attributes of timbre. From this some modifications to the features were developed. The design parameters that must be considered are also discussed.

Different attempts to characterize monophonic and polyphonic timbre are discussed. Finally, related works on the application of timbre for music computing are presented.

Chapter 3 – Timbre similarity. This chapter begins by describing the experimental methods used throughout this work. The audio collections and evaluation measures are discussed. Understanding these measures is important as it affects the applicability of any conclusions. A range of four collections have been used. This helps identify trends in the results. Optimization of the individual features was performed. From the results, we were able to narrow down the best combination of features.

Chapter 4 – Towards efficient similarity estimation. The issue with the candidate system in Chapter 5 is that it is not optimized for large collections. The computational times of the similarity functions does not scale well for large collections. The suitability of the filter-and-refine method is investigated in this chapter. This scheme has the property that a large proportion of the data is discarded so only a few exact similarity computations are performed. This chapter concludes with the results of our submissions to the 2013 MIREX audio music similarity task.

Chapter 5 – Automatic audio classification of audio. The application of our timbre similarity systems are extended to automatic audio classification. In this paradigm, an unlabeled song is tagged with the label of the nearest song. In our local experiments, the nearest neighbour approach is compared with the support vector machine and the Gaussian mixture model. Finally, the performance of our audio classifier is validated in the annual Music Information Retrieval Evaluation eXchange (MIREX). The MIREX results show that our techniques are state-of-the-art methods.

Chapter 6 – Conclusions. This chapter draws some conclusions from the work and ideas in this thesis, highlighting both successes and limitations. Recommendations for future research to extend the usefulness of the proposed system are set out.

Chapter 2: The Role of Timbre in Music Search

This chapter focuses on timbre and its role in music search. Timbre is the property of sound that enables a listener to distinguish one instrument from another. When modelled properly, timbre can be useful for searching music with similar textures. This chapter begins by describing the task of music search. Then, a generic framework for a content-based audio retrieval system is described, explaining each component's role extensively. It is followed by a description of the timbre features used in content-based audio retrieval, including the different perspectives in defining timbre and its attributes.

To compute timbre similarities, it is necessary to extract *features* or *descriptors* from an audio signal. The extracted features should be able to capture the salient attributes of timbre as the audio retrieval system can only be as good as the features it analyses. Considering the attributes of timbre, the following features are extracted: 1) Mel-frequency cepstral coefficients to model the spectral envelope, 2) spectral contrast to describe the range between tonal and noise-like character, 3) sub-band flux to describe the temporal unfolding and shaping of sound spectra and, 4) other spectral features commonly used in the literature. This is followed by an overview of related studies on monophonic and polyphonic timbre. Studies that provided the groundwork for developing computational models of polyphonic timbre are discussed. Finally, research challenges which served as motivations for this study are enumerated.

2.1 The task of music search

This section gives an overview of the task of music search. It begins by describing the behaviour of people that use the web for music search, including common trends and their inadequacies. Next, a background on the field of music information retrieval is given to provide context to discussions later in this chapter. Lastly, different approaches in estimating audio similarity, particularly content-based methods, are described.

2.1.1 Search engine-based music search

The study by Cunningham and Bainbridge (2010) explored music search behaviour by identifying music-related queries in a publicly released set of AOL search logs. The logs consist of over 21 million web queries from over 650,000 distinct users, over a period of three months (March 1-May 31 2006). Music-related queries are defined as queries in which either terms in the query string or elements of the destination URL show that the user is searching music, a song, lyrics, a music representation that will support performance, or a common music audio file format. According to their study, approximately 15% of users conduct at least one music search in the time period studied, and approximately 1.35% of search activities are connected to music.

The analysis of searches over the complete AOL log reveals that the average number of terms per query for the log as a whole is 2.83. Meanwhile, the average number of terms per query for music-related searches is 5. The reason is that audio search queries usually contain terms drawn from song titles, CD, and artist names. Queries of length one is assumed as attempts to locate general sources such as lyrics database or source of mp3 files. Queries of length one to two are usually specific songs and lyrics queries, whereas one to three term queries indicate a desire for general music resource, genre or format.

The study tabulated the 80 most frequently occurring terms in the query statements. Searches on *download* and *free* are strong indicators of a desire to get audio rather than other formats of music information, e.g. bibliographic details, lyrics. The commonly searched music genres include *country*, *gospel*, *rock*, *Christian* and *dance*. The term *video* is also on the list, evidence that music searches can sometimes be satisfied by other multimedia, e.g. a music audio search by the appropriate music video.

For the log as a whole, there are over 92 click-through events for every 100 queries. The ratio is lower for music-related queries: just over 64 selections for every 100 queries. This high failure rate for the identified music-related queries highlights the need for specialized music search applications. The review of (Downie 2005) indicates that keyword search is

ill-suited to satisfy many music information needs. The development of content-based Music Information Retrieval (MIR) systems promises to address this gap.

2.1.2 Music information retrieval

Music information retrieval (MIR) is an emerging field of study dedicated to users' music information needs. It is a very dynamic and active research area as more works are published in the Proceedings of the annual International Society for Music Information Retrieval (ISMIR). MIR research is also increasingly published in high-standard scientific journals and international conferences (Polotti and Rocchesso 2008). One of the pioneers (Kassler 1966) defined MIR as "the task of extracting, from a large quantity of data, the portions of that data with respect to which some particular musicological statement is true." In 1999, the International Symposium on Music Information Retrieval was born thanks to the efforts of Downie, Byrd and Crawford (Downie et al. 2009). Since then, the international MIR research community has experienced significant growth. (Casey et al. 2008) identifies three main audiences that will benefit from MIR research: industry bodies engaged in recording, aggregating and disseminating music; end users who want a more personalized way of finding music; and music professionals that include music performers, teachers, musicologists, copyright lawyers, and music producers.

Music information retrieval is a multi-faceted field that covers, but is not limited to, the following tasks (Fingerhut 2004):

Computational methods for classification, clustering, and modelling — musical feature extraction for mono- and polyphonic music, similarity and pattern matching, retrieval

Formal methods and databases — applications of automated music identification and recognition, such as score following, automatic accompaniment, routing and filtering for music and music queries, query languages, standards and other metadata or protocols for music information handling and retrieval, multi-agent systems, distributed search)

Software for music information retrieval — Semantic Web and musical digital objects, intelligent agents, collaborative software, web-based search and semantic retrieval, query by humming, acoustic fingerprinting

Music perception, cognition, affect, and emotions — music similarity metrics, syntactical parameters, semantic parameters, musical forms, structures, styles and genres, music annotation methodologies

Researchers in MIR usually work on a specific task, e.g. music similarity estimation, music mood classification, instrument recognition, etc. It is through collaboration and understanding between related fields that solutions will be forged out and create meaningful applications. In 2008, the significant challenges for content-based music information retrieval were identified (Casey et al. 2008).

1. Scaling content-based similarity to millions of tracks
2. Integration of tools and MIR frameworks
3. Content description of polyphonic music
4. Addressing the semantic gap
5. User preference modeling
6. User focus

Since then, issues on scalability, user preference modelling, and user focus have been addressed that allowed implementation of MIR techniques on commercial music applications. For example, Shazam[8] , The Echo Nest[9], and Last.fm[10] are commercial MIR systems that work on databases that contain millions of tracks. Shazam is a service that enables identification of a song using recorded audio clip as a query. The Echo Nest is a music recommendation service that combines acoustic and textual analysis. Last.fm is also a music recommendation system that builds a profile of each user's musical taste based on listening patterns.

[8] http://www.shazam.com/music/web/about.html
[9] http://the.echonest.com
[10] http://www.last.fm

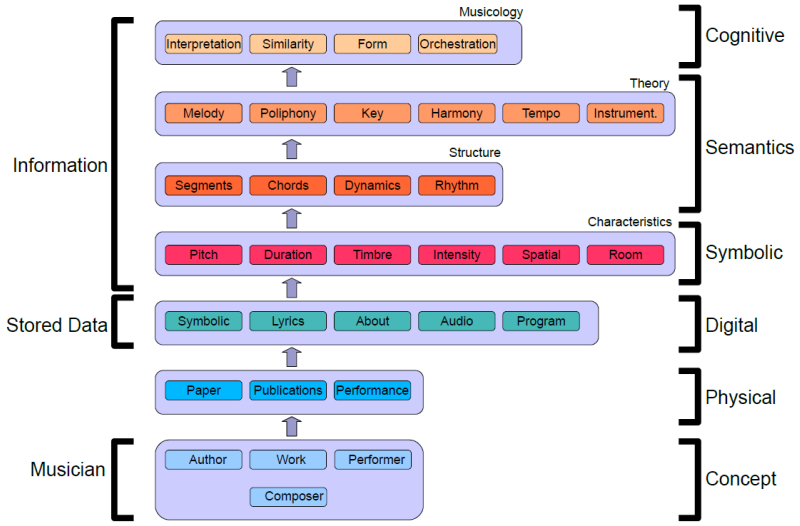


FIGURE 2.1: Simplified version of the MIR map (Fingerhut 2004)

Other MIR issues remain to be addressed. The MIR community has created significant number of tools for audio analysis (e.g. MIRtoolbox (Lartillot and Toiviainen 2007), Sonic Visualiser[11], jAudio[12]) and frameworks (e.g. Marsyas[13]). Cooperation is necessary to standardize these tools and make it easier to develop new systems and applications.

Two remaining MIR issues are the most difficult to resolve. First, polyphonic music is still modelled *globally* rather than identifying the different sound sources that make up the music. Second, a semantic gap still exists where objective measurements of audio features and the subjective description of musical perception remain unresolved.

Michael Fingerhut presented a map of MIR (Fingerhut 2004). FIGURE 2.1 shows a simplified version of this map. The figure shows the related ideas from the musician's mind up to the information from a musicologist's point of view. It also describes the level of features that can be extracted from a musical piece.

For example, a composer has the concept and creates the musical notation for his new single using software. The musical notation is printed in physical form and used for

[11] http://www.sonicvisualiser.org
[12] http://jaudio.sourceforge.net
[13] http://marsyas.info

performance by the chosen artist. The artist's interpretation is stored in digital form. After digital mastering, the single is released in MP3 form with metadata on its performing artist and music genre. The recording can be characterized at various levels by segmenting the audio waveform. Segmenting the signal into very short frames (e.g. 23 milliseconds), signal processing techniques can be applied to extract symbolic representation of pitch, duration, intensity, etc. Audio clips lasting for several seconds make up the segments of the track, e.g. intro, verse, chorus, outro. Each segment can be described by its chords, rhythm, and dynamics. The whole track is characterized by its key, tempo, melody, etc. The major challenge for MIR systems is to infer meaningful information at the cognitive level using symbolic and semantic information.

Music information retrieval has an active research community. Each of the areas described in this section is being studied and evaluated. Performance evaluation is a complex task that is often complicated with copyright issues. Researchers tend to evaluate their systems using their own database. This makes it hard to compare different systems on a level field. To solve this, the International Music Information Retrieval Systems Evaluation Laboratory (IMIRSEL) in the Graduate school of Library Information Science at the University of Illinois conducts annual evaluations of many Music Information Retrieval (MIR) algorithms. Different MIR tasks work on a particular database and performance metrics. The annual evaluations are known as the Music Information Retrieval Evaluation eXchange (MIREX) (Downie 2008). MIREX has become an important indicator of state-of-the-art MIR systems.

One of the important evaluation tasks in MIREX is Audio Music Similarity and Retrieval (AMS)[14]. The evaluation of this task includes objective results from statistics and subjective results from listeners' evaluations. Subjective results from this task showed some inconsistencies of judgments among graders. It is argued that the differences observed can be attributed to the problems in the definition of the evaluation task (Jones et al. 2007). The loose concept of audio similarity results in a variation in

[14] http://www.music-ir.org/mirex/wiki/2013:Audio_Music_Similarity_and_Retrieval

observer scores. Hence, the AMS task may need to be more clearly defined to include more objective features, e.g. melody, instrumentation, style, etc.

In 2012, the Million Song Dataset Challenge has been introduced for offline evaluation of a music recommendation system (McFee et al. 2012). It is the first large-scale, personalized music recommendation challenge where the goal is to predict the songs that a user will listen to. Any type of algorithm can be used: content-based methods, collaborative filtering, web crawling, etc. The challenge provides an opportunity for the MIR research community to test their systems on an industrial-size database and to merge the results from different sub-fields into one system for one specific task.

2.1.3 Music search approaches

Suppose you were in a café earlier and you hear this great tune. You want to learn more about this tune, and possibly get a copy too. So, you immediately grab your smartphone and open Shazam application. Shazam listens to the clip and in less than 10 seconds, returns the related metadata such as album name, artist, music label, and genre. It gives you the option to purchase the song on iTunes. After listening to the whole record, you decide to buy and download the song. The system described in this scenario would not have existed if not for the incorporation of MIR techniques.

The rapid growth in the quantity and accessibility of digital music provides motivation for research into methods for music search and retrieval. At the heart of this task is designing algorithms for music similarity estimation. However, the definition of music involves two aspects: the *qualitative* question of "in what way are two things similar?" and the *quantitative* question of "to what degree are two things similar (Orpen and Huron 1992)?" Thus, any qualitative notion of similarity may be entertained.

Ideally, music similarity estimation algorithms should mimic the human perception of similarity between two tracks. The estimate is produced as one-dimensional metric or scalar values to quantify similarity between tracks. However, human perception is multi-dimensional and music similarity can be described in terms of timbre, rhythm, melody,

structure, tempo, etc. The concept of music similarity is also subjective as it depends on the listener's experience, cultural background, and preference. Hence, the similarity algorithm must clearly define which dimensions of music are considered for estimating similarity. The methods for music search can be broadly categorized into three groups: 1) based on textual data/tags, 2) based on symbolic data such as description of notes in a MIDI file, and 3) based on the content of the music.

The music similarity metric has been implemented in several ways (West 2008):

1) Collaborative filtering (users that played X also played Y)
2) Social tag data (comparing tags applied to tracks by users)
3) Expert metadata (applied to a track by experts using criteria)
4) Symbolic melodic similarity (MIDI format)
5) Direct analysis of audio content

Among these, metrics based directly on analysis of audio content have several advantages:

- They do not require a number of listeners to have listened or tagged a track that leads to 'cold start' problems, i.e. unpopular tracks will almost never be recommended.

- They do not require hours of human labour to create and apply metadata to each track considering that more than a million digital tracks are added annually.

- They are unbiased unlike humans where the metadata applied are either not suitable, or unreliable, or not given at all.

- They can work on a variety of digital music formats such as MP3, wav, etc.

The content-based approaches have been the focus of most research work on MIR. The main idea is that a music file can be described by a set of features directly extracted from its waveform. The processes used to implement music audio content analysis and comparison algorithms are computationally intensive and often scale linearly. Hence, it is important that the algorithms can handle up to millions of tracks while providing a satisfactory level of performance. In Chapter 4, the problem of estimating timbre similarity efficiently is considered. Although not tested with a million-track dataset, the applied techniques have the potential to handle such datasets efficiently.

2.1.3.1 Collaborative filtering based approaches

Automated collaborative filtering systems work by collecting human judgments (ratings) for items in a given domain and matching together people who share the same information needs or the same tastes (Herlocker et al. 1999). Users of a collaborative filtering system share their personal judgments and opinion for each item they consume to guide other users on which items to consume. In return, collaborative filtering systems give recommendations to relevant items that a user browses.

The main advantage of collaborative filtering is that its recommendations are based on human taste or judgment that is very hard for computers to extract. In the case of music recommendation, collaborative filtering has several drawbacks. First, 'cold start' problems exist where a track cannot be recommended until it has been sufficiently played and rated. This prevents the system from recommending new or unknown music. Second, odd recommendations may occur caused by diverse opinion on tracks. This may cause difficulty in finding similar music that a user expects. Third, a collaborative filter assumes that those who agreed in the past will also agree in the future (Kautz et al. 1997).

Nonetheless, collaborative filtering has been utilized successfully by online channels such as Amazon[15], eBay[16] and iTunes[17]. Music information retrieval and the applications in this report aim to complement this approach in providing better experience to listeners.

2.1.3.2 Human-edited metadata based approaches

One of the popular websites based on human-edited metadata for browsing music is Allmusic.com[18]. Its editorial staff and expert contributors made its online service one of the most comprehensive reference sources for music. As of May 2013, their database contains 3 million albums with 30 million tracks worldwide (Rovi 2013). The relational

[15] http://www.amazon.com
[16] http://www.ebay.com
[17] http://www.itunes.com
[18] http://www.allmusic.com

links provide meaningful connections between artists and their music such as major influences, similar artists and followers. On top of that, descriptive (styles, moods, instruments. etc.) and editorial content (biographies, album and song reviews) are given. Allmusic.com has an independent editorial team and is one of the most respected and trusted sources by consumers and industry professionals in music content and criticism. Its service received good reviews from consumers and remains one of the most accessed online music information channels[19].

The metadata-based scheme, such as Allmusic.com, has its drawbacks. First, a set of musically relevant attributes must be defined to label the catalogue. This takes significant amount of work among editors to make the attributes as complete as possible. Second, thousands of albums are released each year and some less-known music will take longer time to be indexed. This results in a 'long tail' in the catalogue wherein popular artists dominate while unpopular or new artists are less likely to be recommended. Third, the relational link (e.g. similar artists) assigned is very subjective and may be biased based on the musical background of the editors.

The social media web addresses the limitations of centralized metadata by opening the task of describing content to the public. Communities of users apply tags that they think are appropriate to the music tracks. Common classes of tags are *mood*, *emotion*, *genre*, *style*, etc. Last.fm is an example of a music portal that combines collaborative filtering with user tagging. However, such service suffers from the diversity of users that may affect the reliability of the descriptions.

2.1.3.3 Audio content-based approaches

In the 20th century, the development in technology and research in artificial intelligence paved way for a computational approach to understanding music. The goal was to create systems that could model human perception of music, e.g. recognize musical structures like melodies, rhythm, etc. at the same level of judgment as humans. Although there has

[19] http://www.alexa.com/siteinfo/allmusic.com

been some success in specialized problems such as algorithmic composition of music (Edwards 2011), complex musical processes are still outside the range of computers. For example, no system is currently capable of identifying and separating an arbitrary number of instruments in polyphonic (simultaneous instruments) music.

The new research field of music information retrieval is taking a more practical point of view. Its main goal is the provision of a level of access to the world's vast store of music on a level equal to, or exceeding, that currently being afforded by text-based search engines (Downie 2008). The real goal is not so much for a computer to understand music the way humans do, but rather to have enough intelligence to support musical services and applications to users. However, textual analysis such as metadata comparison and mining social tags are also important aspects of music information retrieval.

Research work done on music information retrieval involves a combination of direct audio content analysis, textual analysis of metadata and web-crawled data, and user modelling (Casey et al. 2008). The main idea behind content-based approaches is that an audio signal can be described by a set of *features* or *descriptors* that are directly computed from the waveform. Specific methodologies are required to extract a particular feature from a particular medium. The algorithms developed for *feature extraction* are mostly derived from signal processing, psychoacoustics, statistics, musicology and probability theory. This enables MIR systems to automatically analyze and understand the contents of musical pieces even without the metadata. As compared with other approaches for music similarity estimation, content-based approach is objective and treats all pieces of music equally.

Given all its advantages, content-based approach to music search is chosen for implementation in this thesis.

2.2 Basic content-based query system overview

The basic components of a content-based query system are shown in Figure 2.2. Given a query song, features are extracted from the waveform. The features of the query song are

matched to the music features of other songs in a database using a similarity algorithm. The match can be exact, retrieving documents with specific content, or approximate, retrieving near neighbours in a musical space where proximity encodes musical similarity. The songs in the database are ranked according to their similarity. The system then returns a list of songs that are predicted as similar to the query song. The speed of retrieval depends on the computational complexity of the algorithms and the indexing method used.

The key components of the system are the feature extraction and matching as they contain the audio information and similarity metrics. The indexing and accessing methods also depend on them. The following sections describe each block.

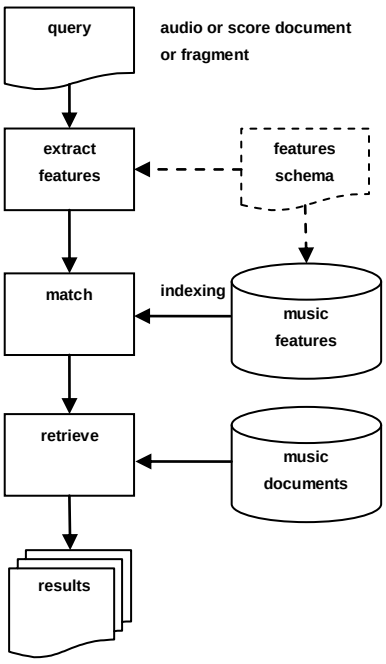


FIGURE 2.2: Flowchart of basic content-based query system (Casey et al. 2008)

2.2.1 Query

Music information retrieval systems may perform several types of queries for a given music database. The first is *query-by-example* that finds music similar to a given audio clip. The second is *recommendation queries* that find music similar to audio files already owned. The third is *playlist generation queries* that find music similar to a given number

of tracks. There are other options in the queries but all of these require music tracks in digital format and the estimation of music similarity between audio tracks. The obvious limitation of using a music file as query is where to get it from if you do not own it. Aside from purchasing online, there are two alternatives to overcome this problem: 1) query by singing or humming, and 2) audio fingerprinting.

Query by singing or humming systems take a user-hummed melody and compares it to their database. The system then returns a ranked list of music closest to the query. SoundHound[20] for mobile devices and Midomi[21] for online queries are commercially-available *query by humming* services. Although fun to use, in reality these systems greatly depend on the quality of the input query. For example, a completely out-of-tune input query might probably result in mismatches. Moreover, it can only return tracks with the same melody.

An *audio fingerprint* is a condensed digital summary, deterministically generated from an audio signal, and can be used to identify an audio sample in an audio database[22]. Gracenote[23] and Shazam[24] provide such service in mobile devices that lets users identify songs by simply holding their phone to the music source. These applications identify the song by analyzing the input audio clip and matching it to its unique audio fingerprint in their database. They can also return the associated song metadata and even the cover art of the album. The rich metadata (genres, artist types, origin, and era) can then be used for suggesting new music, e.g. iTunes Genius.

---

[20] http://www.soundhound.com

[21] http://www.midomi.com

[22] ISO IEC TR 21000-11 (2004), *Multimedia framework (MPEG-21) -- Part 11: Evaluation Tools for Persistent Association Technologies*

[23] http://www.gracenote.com

[24] http://www.shazam.com

2.2.2 Features

The content-based query system should produce a ranked list of music tracks that are relevant to the query. Therefore, to rank music tracks, we must first extract a quantitative description of what we want to compare. These are the audio features that are at the core of content-based query systems.

2.2.2.1 High-level music content description

By intuition, access and retrieval of music should focus on music content, in terms of music features (Orio 2006). There are several dimensions based on music theory and analysis that may be used separately to describe a musical work. The dimensions of music that could be effective for MIR systems are the following:

- **Timbre** depends on the perception of the quality of sounds, which is related to the musical instruments used, possible audio effects, and to the playing techniques. All the musical gestures contribute to the perception of the overall timbre of a performance. This is the dimension that enables a listener to distinguish one instrument from another. A detailed description of timbre is given in Section 2.3.

- **Orchestration** is due to the composers' and performers' choices in selecting which musical instruments are to be employed to play the different voices, chords, and percussive sounds of a musical work.

- **Acoustics** can be considered as a specialization on some characteristics of timbre, including the contribution of room acoustics, background noise, audio-post processing, filtering and equalization.

- **Rhythm** is related to the periodic repetition, with possible small variants, of a temporal pattern of onsets alone. Unpitched and percussive sounds are the most used conveyors of the rhythmic dimension.

- **Melody** is made of a sequence of tones with a similar timbre that have a recognizable pitch within a small frequency range. The singing voice and monophonic instruments that play a similar register are normally used to convey the melodic dimension.

- **Harmony** is the organization, along the time axis, of simultaneous sounds with a recognizable pitch.

- **Structure** is a horizontal dimension whose time scale is different from the previous ones, being related to macro-level features such as repetitions, interleaving of themes and choruses, presence of breaks, changes of time signatures, etc.

Even though these dimensions are intuitive to listeners, the tasks of extracting these dimensions remain extremely difficult to achieve. Hence, extraction of high level music content descriptions is an area of extensive research.

2.2.2.2 Low-level audio features

Aside from metadata and high-level content description, audio features are usually derived from the information in the digital audio. Low-level audio features contain information about a musical work and try to model certain aspect or aspects of music. In general, low-level features are derived from segmented audio. The basic segmentation can be done in two ways: frame-based segmentations (periodic sampling with or without overlap at 10-1000 ms intervals) and beat-synchronous segmentations (features are aligned to the estimated musical beat boundaries, i.e. aligned to a series of identical yet distinct periodic short-duration stimuli perceived as points in time, called *tactus* (Orpen and Huron 1992)). The key assumption in working with segmented audio or *frame* is that the signal can be regarded as stationary over short time intervals. A window function (e.g. Hamming or Hanning window) is applied to each frame to minimize spectral leakage. Consecutive frames may be analyzed with some overlap (usually up to 50%) for smoother analysis. Some of the commonly derived low-level features in the literature are enumerated below. The low-level features used in this thesis and their corresponding mathematical descriptions are described in Section 2.3.

- **Short-time magnitude spectrum** – The discrete Fourier transform (DFT) (Oppenheim and Schafer 1989) converts each audio frame from the time domain into the frequency domain. Both the magnitude and phase are produced by the DFT but only the magnitude is usually used (Jensen et al. 2006).

- **Constant-Q/Mel spectrum** – a filter bank is applied to the magnitude spectrum to model the human ear's response. The ear's response is logarithmic in frequency and uses non-uniform bandwidth, known as *critical bands*. Critical

band refer to the frequency bandwidth of the auditory filter within which a second tone will interfere with the perception of the first tone by auditory masking (Fastl 2005). A constant-Q system uses filters whose ratio of bandwidth to center frequency is equal (Brown and Puckette 1992). In contrast, Mel spectrum has linearly spaced filters in the lower frequency range and logarithmically spaced filters above 1300 Hz (Stevens et al. 1937).

- **Mel-Frequency cepstral coefficients** – takes the logarithm of the Mel magnitude spectrum and decorrelate the resulting values using a Discrete Cosine Transform (Rabiner and Juang 1993). The resulting values are the shorthand representation of the energy distribution across the frequency bands. The MFCCs is one of the main features used in this research. Please refer to Appendix 1 for the mathematical descriptions and steps for extracting MFCCs.

- **Spectral flux** – is a measure of how quickly the power spectrum of a signal is changing. It is usually calculated as the 2-norm between two normalized spectra (Tzanetakis and Cook 2002). The spectral flux can be appended to MFCCs to improve timbre description (de Leon and Martinez 2012c).

- **Pitch-class profile (Chromagram)** - uses frequency folding to represent the energy due to each pitch class in twelfth-octave bands (Fujishima 1999). There are 12 equally spaced pitch classes in Western tonal music, so there are typically 12 bands in a chromagram.

- **Onset detection** – is concerned with marking the beginning of notes to separate musical events. This is important in deriving other features such as tempo (Klapuri 1999).

- **Tempo (Beat tracking)** – can be derived from onset detection, and it is often used to align the other low-level audio features (Dixon 2001, Alonso and David 2004).

Low-level features in themselves are insufficient to describe musical concepts as they encode information for very short segments of music. Therefore it is necessary to collect audio frames into one of several aggregate representations. Some of the common types of aggregation are state models and bag-of-frames models. The state models have a fine spectral-temporal structure, meaning spectral information is mapped to time, hence preserving temporal information. The Hidden Markov model is a good of example of this. The state-model is applicable for more selective searches such as fingerprinting and cover song identification(Cano et al. 2005). The second type of aggregation is the bag-of-frames

approach that models global statistics. The Gaussian mixture model or single Gaussian is often used for this (Aucouturier and Pachet 2002, Mandel and Ellis 2005). The bag-of-frames approach is more applicable to less selective searches like similar artist or genre retrieval. Aside from Gaussian mixture models, other methods of modelling timbre features are discussed in Section 2.3.3.

2.2.3 Match and Retrieve

The next step after extracting relevant features from the input query is to compare these features with the music database. For humans, the concept of music similarity depends on their interpretation of the music and the context in which it is played (Jones et al. 2007). To understand the audio similarity estimation task, let us look at Figure 2.3, adapted from Casey and Slaney (2006). It illustrates the specificity spectrum, from exact to near neighbours to a more global matching. The upper text shows the retrieval tasks and the lower text shows the aggregation approach for low-level features. Notice that audio semantic gap exists in between the extremes of the spectrum. The tasks on the left of the spectrum require matching of specific audio content (high-to-mid specificity systems) and those on the right require matching of high-level music concepts (mid-to-low specificity systems).

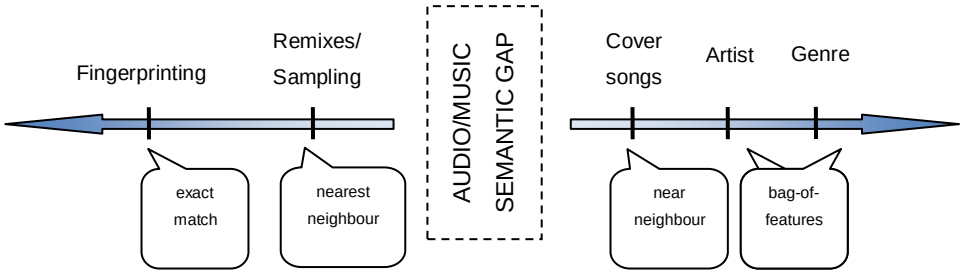


FIGURE 2.3: Audio similarity specificity spectrum (Casey and Slaney 2006)

In research literature, most common music similarity measures use the timbre component. The main reason is that music from the same genres typically sounds the same, e.g. same instruments, acoustics, etc. The second reason is timbre modelling has

long been studied in the field of speech recognition. Many of the techniques used in speech recognition were adapted to music information retrieval. Logan and Salomon (2001), as well as Aucouturier and Pachet (2002) were among the pioneers to model timbre using the Mel-frequency cepstral coefficients (MFCCs). They used musical genre classification accuracy to evaluate their music similarity measure. The music genres used were classical, country, disco, hip-hop, jazz, and rock. These western music genres have a distinct sound and style that can easily differentiate one from the other. Currently almost all music similarity systems use timbre component as the main feature.

Another music dimension that has been explored for music similarity estimation is rhythm. In general, rhythm can be thought of as the average tempo with which listeners would tap their foot while listening to the same piece of music. In some studies, it is possible to use a representation that provides saliency values of rhythm then use it for music genre classification (Tzanetakis and Cook 2002, Pampalk et al. 2005, de Leon and Martinez 2012c). Their studies showed that combining timbre similarity with rhythm similarity measures improved the average genre classification performance. The rhythm component is also useful for non-western music that is often characterized by a certain repeating pattern. Some examples include Indian (De and Roy 2012) or Chinese (Liu et al. 2008) music.

In other applications, for example in cover song identification or audio fingerprinting, it is preferred to have a representation that is related to the pitch content of the music rather than the orchestration and voices that are playing. This would be easy for a song that has a fully transcribed music score. However, there is no system that can automatically transcribe any music file with reliable results. Instead, the pitch content representation is measured based on the occurrences of specific discrete musical pitches in a music segment. Usually, the pitch profile is based on the Western 12 pitch classes. These features were used for cover song identification (Ellis and Poliner 2007, Serra and Gomez 2008).

Recent studies have considered music mood or emotion detection for music search. Like the notion of music similarity, mood or emotion detection is subjective and it depends on

many factors including the listener's background and present disposition to music listening (Juslin and Laukka 2004). The study by Krumhansl (2002) investigated the dynamic aspect of musical emotion and its relation to the cognition of musical structure. The research suggests that musicians vary attributes to express different emotions. Automatic music mood/emotion detection and classification using raw audio signal has been done by Liu et al. (2003) and Yang et al. (2006). The basic moods used are based on the two dimensional positiveness-arousal diagram from Russell (1980).

The deterministic similarity functions used in music information retrieval do not accurately represent the cognitive nature of human similarity. In fact, Tversky (1977) argued that a geometric representation of similarity, such as a metric similarity function, has little relation to perceptual similarity. Human perception does not obey the metric properties of symmetry and triangle inequality. For example, track A is perceived similar to track B because they have similar melody; track B is perceived similar to track C because they have similar orchestration; it does not necessarily imply that track A is similar to track C. However, the geometric representations of similarity have been exploited in content-based query systems to combine feature spaces and apply efficient similarity estimation methods.

Once the feature or features to be used for music similarity estimation is identified, the specificity of retrieval can be determined. The match can be exact, retrieving documents with specific content, or approximate, retrieving near neighbours in a musical space where proximity encodes musical similarity.

2.2.4 Indexing

Efficient indexing of the feature database is an important issue that limits the integration of content-based query systems in commercial applications. The direct approach to a content-based search is to compare the features of the input query to the features database then find its nearest neighbours. If we perform a full linear scan of the database, then the complexity of the task is *O(n)*. In this respect, we can say that the content-based system is scalable. However, the potentially large number of music tracks, the

dimensions of features and the computational complexity of similarity computations severely affect search and retrieval time. This means that it would still be very difficult to achieve real-time searches even with a database containing only thousands of tracks. To improve scalability, a faster indexing method is required. This may come from limiting the search space where the linear scan is performed, or developing an indexing method with a lower computational complexity.

Indexing methods have been studied and successfully applied in related fields like text retrieval. Unfortunately, these methods cannot be simply integrated in content-based query systems. The feature vectors extracted and similarity functions are usually high-dimensional and the consequence of this, often called as the *curse of dimensionality*, makes traditional indexing ineffective. Consider the case of searching for k-nearest neighbours for a given query. As the number of dimensions increases, the distance values from the nearest neighbours become more similar. This results in the emergence of *hubs*, which are items that appear more often in nearest neighbours search from the dataset than expected (Radovanović et al. 2010).

2.2.5 Ranking and results

The ranking of results can be a weighted linear combination of distances computed using several features. The results are presented as a list where the user should then be able to browse and listen to the music clips. An alternative is to graphically organize similar pieces on a two-dimensional map to create *Islands of Music*, see Figure 2.4 (Pampalk et al. 2002).

If we consider the task of audio retrieval it becomes obvious that it is important to return only the top few nearest neighbours. Unlike text or image items, music files are usually previewed one at a time for several seconds. For someone searching a database with more than a million tracks, it would be more practical to return only the top ten to hundred rather than a ranking of the whole database. This idea leads us to focus on the behaviour in the vicinity local to the query.

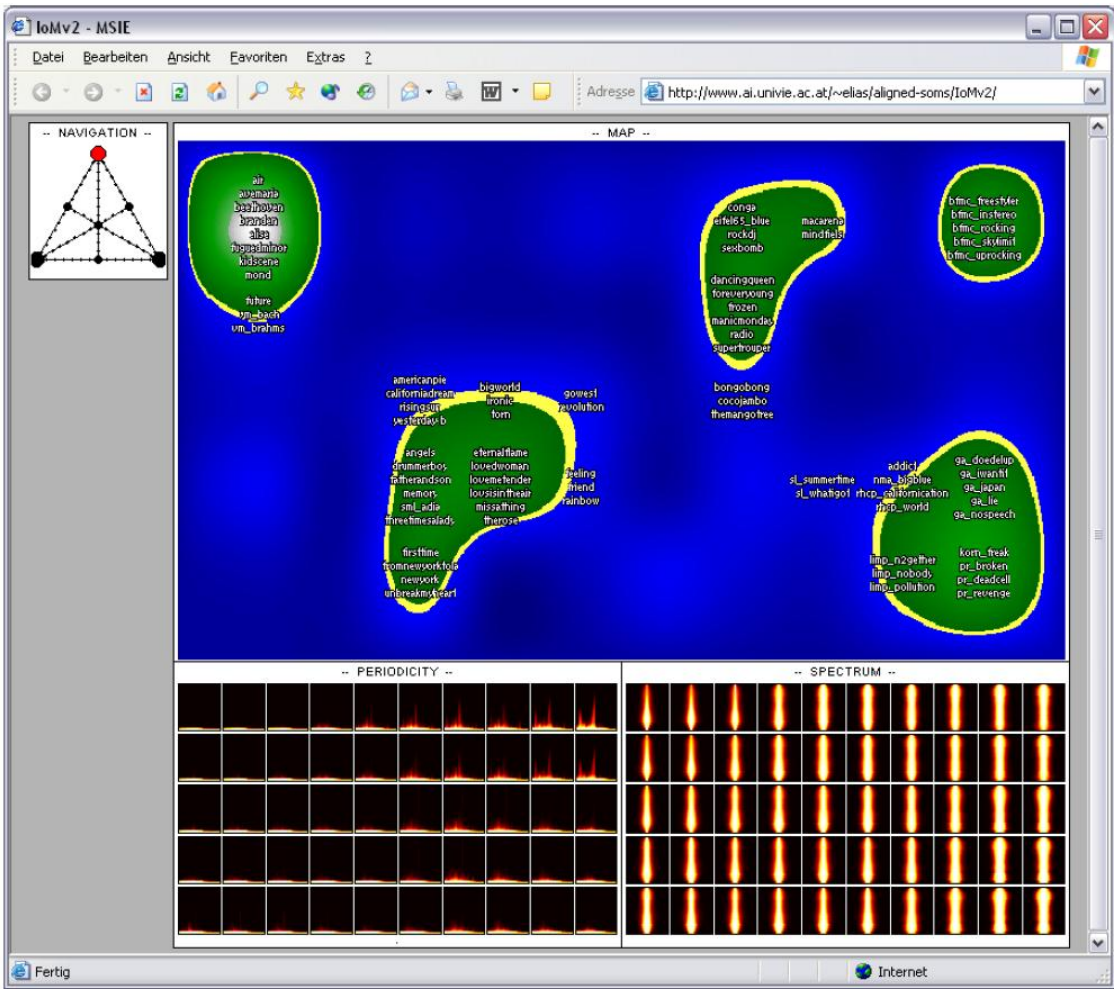


FIGURE 2.4: Screenshot of Islands of Music database browser, (Pampalk et al. 2003)

2.3 Timbre features for content-based audio retrieval

The performance of a retrieval system can only be as good as the features it uses. MIR researchers are keen on identifying the best features that suit their application. Given the available tools for MIR research, it is possible to derive many features without understanding their meaning. However, this approach hinders the potential to optimize the system by selecting only the relevant features and tuning the appropriate parameters.

The features are important because they provide a simplified representation of the data. If the properties that a user is interested in are not captured by the features, then even the most complex system will not be able to satisfy the user's requirements.

Most musical instruments, with the exception of some percussive instruments like cymbals or snare drums, produce almost periodic vibrations. Specifically, the sounds produced by musical instruments are the result of the combination of a *fundamental frequency* and higher frequency components called *overtones*. A musical sound can be described by its pitch, loudness, and timbre (Orio 2006). The pitch is related to the perception of the frequency. The frequency range of hearing for pure tones spans from 20 Hz to 20 kHz. The loudness is related to the energy of the sound vibration and its perception is dependent on frequency. In the frequency range between 2 and 5 kHz, the minimum sound energy level to perceive sound, called *threshold of quiet*, is on the vicinity of 0 dB. For the same frequency range, the sound energy level that can cause pain, called *threshold of pain*, is around 130 dB (Fastl 2005). Timbre is the characteristic that allows listeners to differentiate between two sounds with the same pitch and loudness. Unlike pitch and loudness, timbre is a multidimensional sound quality that cannot be easily described with simple features.

This section focuses on timbre as it is the most commonly used music component in MIR tasks such as audio music similarity, genre classification, and emotion classification. This thesis aims to improve the computational model and similarity estimation of timbre. Hence, it is important to understand and appreciate its complexities. This section begins by defining timbre from different perspectives. The different factors that affect the perception of timbre are enumerated. This is followed by an overview of related studies on monophonic and polyphonic timbre. Lastly, studies that provided the groundwork for developing computational models of polyphonic timbre are discussed.

2.3.1 Introduction to timbre

Timbre means different things for different people (Smalley 1994).

1. Many traditional musicians quote the American Standards Association definition as "that attribute of sensation in terms of which a listener can judge two sounds having the same loudness and pitch are dissimilar (ASA, 1960)."
2. For the contemporary instrumental composer, timbre is an extension of harmony, or vice versa. The composer uses spectral analysis information to find a

relationship between pitch and sound qualities, and attempts to negotiate fluent border crossings between the two.

3. For researchers, timbre is composed of multiple variables to determine its identity. Hence, it has become a challenge to differentiate what is acoustically present in sounds and what is psychoacoustically relevant.
4. For ordinary people, timbre is related to the 'matter' of sound. Everyday terms like bright/dull, compact/spread, hollow/dense, are used as qualitative descriptions of sound qualities.

Although the standard definition of timbre is "everything that is not loudness or pitch", it is argued that pitch and timbre should never be presented as independent variables in perception studies (Houtsma 1997). Another view is that timbre depends on the frequency content and the spectral profile of the sound (Fastl 2005). In addition, the temporal envelope of an instrumental sound, including attack, decay and modulation of the steady-state portion, influences the perceived timbre to such an extent that changes on any of them can make the sound of an instrument unrecognizable (Berger 1964).

The complex and ambiguous nature of timbre makes it very interesting resulting in a variety of ways of developing timbre models. The different methods which are commonly used in timbre research can be divided into three broad classes (De Poli and Prandoni 1997): 1) models of the signal, 2) models which take into account underlying sound-producing mechanisms, and 3) models which take into account the properties of auditory perception.

The classical models for the representation of sounds used in the field of timbre research use a time-frequency representation of the signal. The models are based on the short-time Fourier transform (STFT) and other derivative features including spectral shape, spectral centroid, spectral flatness, etc. An audio signal is usually cut into smaller segments called frames, typically between 10 to 50 milliseconds, upon which the signal becomes pseudo-stationary. Thus, the Fourier transform becomes a valid tool in analysing the signal. The visual representation of the frequency components and its evolution in time is called a *spectrogram*, see FIGURE 2.6.

The second type of timbre models is based on the knowledge of sound-producing mechanisms. This approach is employed widely in speech analysis, e.g. source plus linear

system models such as linear predictive coding (LPC). For example, the source in a speech signal is assumed to be produced by the vibration of the vocal chords. The throat and mouth, which can be represented by linear models, form the tube where the sound passes. Application of this type of models to music is unsuitable since the polyphonic timbre is an ensemble of musical instruments with highly nonlinear characteristics.

The third type of timbre models, initially developed by the speech processing community, relies on the characteristics of the acoustic perception. The perceptual behaviour of the human auditory system is taken into account while designing algorithms for signal representation. Because the human ear has limited ability to distinguish between distinct tones, the frequency spectra can often be represented by a vector representing the amount of instantaneous acoustic power in each critical band without much perceptual loss of information (Plomp 1970). The most significant example in attempting to improve acoustic analysis of speech by perceptual related knowledge is given by the Mel-frequency cepstrum analysis of speech (Davis and Mermelstein 1980), which transforms the linear frequency domain into a logarithmic scale that resembles the human auditory sensation of tone height.

All of these sound-processing schemes use "short-time" analysis. To track dynamic changes in sound properties, the analysis frames are sometimes overlapped with each other which increase the number of frames to be processed. Thus, time derivatives such as velocity-type and acceleration-type parameters are also included (Furui 1986). The addition of these temporal changes improves automatic speech recognition (ASR) systems and is now included in their acoustic front-end.

One of the fundamental limitations of an STFT-type analysis is that once the analysis window has been chosen, the time frequency resolution is fixed over the entire time-frequency plane since the same window is used at all frequencies. As a further improvement to overcome this limitation, the Wavelet Transform has been introduced (Rioul and Vetterli 1991). It is characterized by the capability of implementing multi-resolution analysis. The Wavelet Transform of a signal is calculated by passing it through a series of filters. First, the signal is simultaneously decomposed using a low pass

and high pass filter. Since half the frequencies of the signal have been removed, half the samples can be discarded according to the Sampling Theorem (Proakis and Manolakis 2006). The filter outputs are then subsampled by 2. This decomposition halves the time resolution since only half of each filter output characterises the signal. However, each output has half the frequency band so the frequency resolution is doubled. The decomposition is repeated to further increase the frequency resolution.

With this processing scheme, if the analysis is viewed as a filter bank, the time resolution increases with the central frequency of the analysis filters. In other words, different analysis windows are simultaneously considered to simulate more closely the frequency response of the human cochlea. As with the preceding processing schemes, this new auditory-based technique is still based on a mathematical model of the signal from which it tries to directly extrapolate a more realistic perceptual behaviour.

The complexity in modelling timbre lies in its multidimensional attributes. For example, the sensation of pitch can be correlated with the perceived dominant fundamental frequency. Meanwhile, the sensation of loudness can be correlated with sound energy. But there is no single physical measure for timbre. Many commentators have attempted to decompose timbre into component attributes. For example, Schouten (1968) describes the elusive attributes of timbre as determined by at least five major acoustic parameters, namely:

1. The range between tonal and noise-like character.
2. The spectral envelope.
3. The time envelope in terms of rise, duration, and decay.
4. The changes in both the spectral envelope (formant-glide) and fundamental frequency (micro-intonation).
5. The prefix, or onset of a sound, quite different to the ensuing lasting vibration.

(Smalley 1994) suggests that timbre is concerned with the temporal unfolding and shaping of the sound spectra; this is called *spectromorphology*. This agrees with the fourth attribute identified by Schouten. Spectromorphology is concerned with motion and growth processes, which may or may not be sonic phenomena. Since sound is a form

of energy that evolves in time, it follows that the timbre attribute depends on the spectromorphological ensemble.

2.3.2 Modelling attributes of timbre

All features that will be described in this section are derived from a signal's spectral representation. A more limited approach is to use information from the time domain, such as *zero-crossing rate* to detect noisiness of the signal (Tzanetakis and Cook 2002). The advantage of time-domain features is that they can be computed easily and efficiently. However, a more meaningful approach is to analyze the frequency content of an audio signal given that any sound is just a summation of sinusoids of different strengths and frequencies.

2.3.2.1 Spectral envelope

The spectral envelope is a curve in the frequency-amplitude plane that matches the amplitudes of the individual partials of the spectrum (Schwarz 1998). Partials are fractional multiples of the fundamental frequency. To illustrate, FIGURE 2.5 plots the spectrum of a piano sound and its corresponding spectral envelope. The spectral envelope is the basic defining factor for a sound. Hence, sounds with similar spectral envelopes are generally perceived as similar. The Mel-Frequency cepstral coefficients (MFCCs) are commonly used to represent spectral envelope. MFCCs are a compact representation of an audio spectrum originally used to model important characteristics in speech (Davis and Mermelstein 1980).

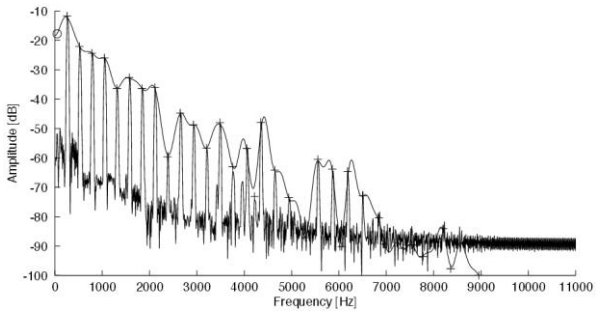


FIGURE 2.5: Spectrum and spectral envelope of a piano sound. Reproduced from (Schwarz 1998)

The MFCCs are the result of a cosine transform of the real logarithm of the short-term magnitude spectrum after it has been passed through a Mel-frequency scale filter bank. The Mel-frequency scale filters are intended to approximate the distribution of the ear's critical bandwidths with frequency, using filters placed roughly linearly at low frequencies and logarithmically at higher frequencies. The important aspects of the human auditory system which MFCCs model are: (1) the non-linear frequency resolution using the Mel frequency scale and, (2) the non-linear perception of loudness using decibel.

The steps for computing the MFCCs for each signal segment are as follows:

1. Calculate the power spectrum using Fast Fourier Transform (FFT).
2. Transform the power spectrum to Mel-scale using a filter bank consisting of triangular filters.
3. Get the sum of the frequency contents of each band.
4. Take the logarithm of each sum.
5. Compute the discrete cosine transform (DCT) of the logarithms.

The complete details on the derivation of MFCCs are described in Appendix 1. Figure 2.6 illustrates the computation steps on an arbitrary audio signal. The first plot shows the time-domain waveform. The second image shows the evolution of the magnitude spectrum with time, also known as spectrogram. The spectrogram has a frequency resolution of 256 frequency bins[25]. The third image shows the spectrogram after applying the Mel-frequency filter banks. The frequency resolution decreased from 256 frequency bins to 36 Mel-frequency bins. The DCT is then applied to obtain the MFCCs, shown in the fourth image, where the 36 Mel-frequency bins are further reduced to 20 coefficients. The MFCCs are able to represent the 512-point magnitude spectrum to just 20 coefficients. The last image shows the reconstructed Mel-frequency spectrogram from the MFCCs. Notice that there is spectral blurring or smoothing compared to the original due to some information lost by applying DCT.

[25] 512-point FFT was used to compute the magnitude spectrum but only half is needed due to even symmetry

Delta and Delta-Delta Coefficients

As the audio signal is dynamic, the MFCCs also change over time. The first and the second instantaneous derivatives of each MFCC can be computed to capture temporal changes. These are called delta-MFCCs (ΔMFCCs) and delta-delta-MFCCs (ΔΔMFCCs) respectively. Delta coefficients are computed using the following formula:

$$d_t = \frac{\sum_{\theta=1}^{\Theta} \theta(c_{t+\theta} - c_{t-\theta})}{2\sum_{\theta=1}^{\Theta} \theta^2} \quad 2.1$$

where d_t is a delta coefficient at frame t, c is the cepstral coefficient, computed using a time window Θ. The same equation can be applied to the delta coefficients to obtain the delta-delta coefficients.

$$dd_t = \frac{\sum_{\theta=1}^{\Theta} \theta(d_{t+\theta} - d_{t-\theta})}{2\sum_{\theta=1}^{\Theta} \theta^2} \quad 2.2$$

Since the MFCCs are static coefficients that depend on the time frame they were derived, the delta and delta-delta coefficients are also known as velocity and acceleration coefficients, respectively.

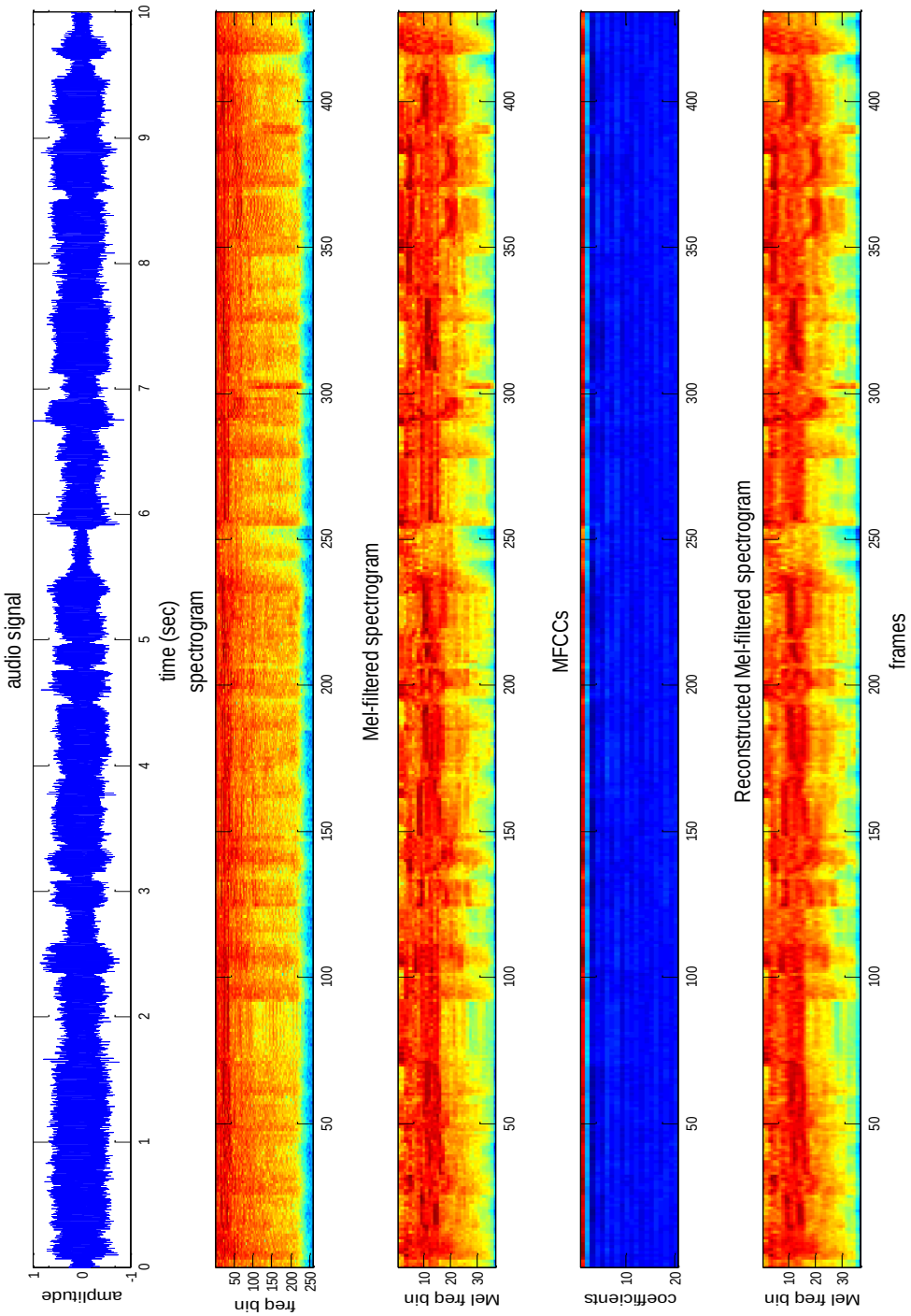


FIGURE 2.6: MFCC computation steps for a test music file. Time is plotted on the x-axis (the temporal resolution is about 23 ms per frame).

2.3.2.2 Modified spectral contrast

An alternative representation of the spectral characteristics of a music clip called octave-based *spectral contrast* has been proposed by Jiang et al. (2002). It considers the spectral peak, spectral valley, and their difference in each sub-band. In music signals, dominant spectral peaks roughly correspond with harmonic components; while non-harmonic components, or noises, often appear at spectral valleys. Hence, this feature models an attribute of timbre on the range between tonal and noise-like character. In deriving MFCCs, the spectral distribution in each sub-band is averaged, and thus loses relative spectral information. The spectral contrast keeps more information and may differentiate spectral distributions that have similar average spectral characteristics.

The spectral contrast algorithm published in Jiang et al. (2002) is very similar to the MFCC algorithm, see Figure 2.7. The authors replace the Mel filter bank used in MFCC analysis by an octave-scale filter bank. An octave is the interval between two frequencies having a ratio 2:1. They decorrelate the spectral contrast coefficients using the Karhunen-Loeve transform. The process of decorrelation aims to reduce the cross-correlation of the signals between adjacent frequency bands due to overlapping filters. In our implementation, we still use the Mel-scale filters to optimize the system since the output of this block is used for MFCC and spectral contrast. We use DCT to decorrelate the coefficients as it is a good approximation of the K-L transform for music signals (Logan 2000).

Raw spectral contrast features estimate the strength of spectral peaks, valleys and their differences in each sub-band. The strength of the peaks and valleys are estimated by the average value in the small neighbourhood around the maximum and minimum values respectively, instead of the exact maximum and minimum value themselves. Suppose the magnitudes of the FFT vector with N frequency bins of the k-th sub-band is denoted as $[M_{k,1}, M_{k,2},\ldots,M_{k,N}]$. After sorting it in descending order, the new vector is represented as $[M'_{k,1}, M'_{k,2},\ldots,M'_{k,N}]$, where $M'_{k,1}>M'_{k,2},\ldots,>M'_{k,N}$. The strength of spectral peaks and spectral valleys are estimated by the average value in the small neighbourhood around maximum and minimum values:

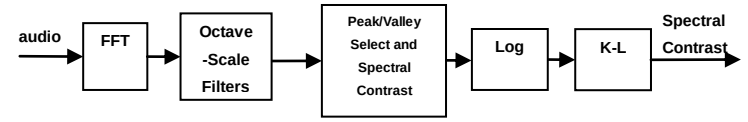
$$peak_k = \log\left(\frac{1}{\alpha N}\sum_{i=1}^{\alpha N} M'_{k,i}\right) \quad 2.3$$

$$valley_k = \log\left(\frac{1}{\alpha N}\sum_{i=1}^{\alpha N} M'_{k,N-i+1}\right) \quad 2.4$$

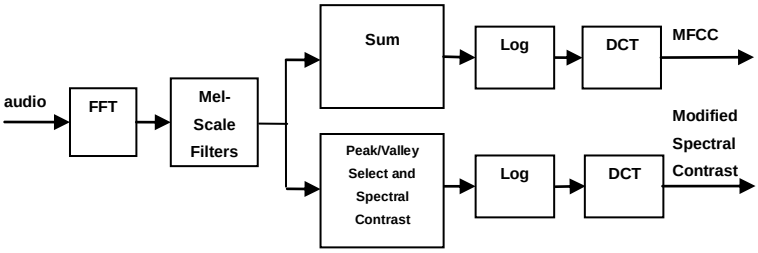
where α is the neighbourhood factor set at 0.2. The neighbourhood factor controls the number of FFT magnitudes to be used in these equations. The corresponding spectral contrast of the k-th sub-band is computed as:

$$SC_k = peak_k - valley_k \quad 2.5$$

The spectral contrast coefficients are concatenated to form a vector. Lastly, the DCT of the spectral contrast vector is computed to get the coefficients.



(a) octave-based spectral contrast



(b) Mel-based spectral contrast

FIGURE 2.7: Block diagrams of the original octave-based spectral contrast (top) and Mel-based spectral contrast (bottom).

2.3.2.3 Spectrotemporal evolution

Sound is a form of energy that evolves in time. Hence, an important attribute of timbre depends on the spectrotemporal evolution. A set of features called *sub-band flux* has been proposed by Alluri and Toiviainen (2009) to represent the fluctuation of frequency content in octave-scaled bands of the spectrum. FIGURE 2.8 shows the block diagram for the calculation of sub-band fluxes. The audio signal is passed through a filter bank to produce 10 channels. The change in energy, or flux, is then computed between adjacent time frames of each sub-band.

In the study by (Alluri and Toiviainen 2009), the authors determined that Mel-flux has lower perceptual coefficients than octave-scale sub-band flux. Hence, we use octave-scale filter banks. The division into sub-bands was obtained using a 10-channel filter bank of octave-scaled fourth-order elliptic filters. The sub-bands are defined as follows: {0 ~ 25Hz, 25 ~ 50Hz, 50 ~ 100Hz, 100 ~ 200Hz, 200 ~ 400Hz, 400 ~ 800Hz, 800 ~ 1600Hz, 1600 ~ 3200Hz, 3200 ~ 6400Hz, 6400 ~ 11025Hz} where the sample rate is 22050Hz. The first sub-band is not perceptually important since the minimum audible frequency is 20 Hz. However, this may help increase the precision of the model. Elliptic filters are a type of digital filter that have a faster transition gain between the passband and stopband than other filters of the same order. The frequency response of the filter bank is shown on FIGURE 2.9.

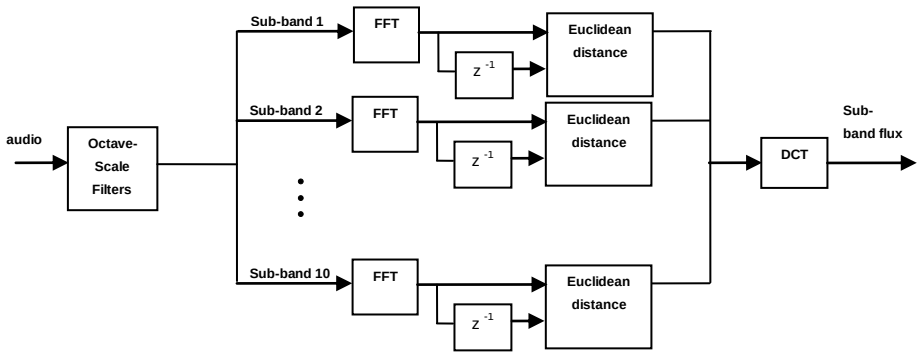


FIGURE 2.8: Block diagram of sub-band flux extraction.

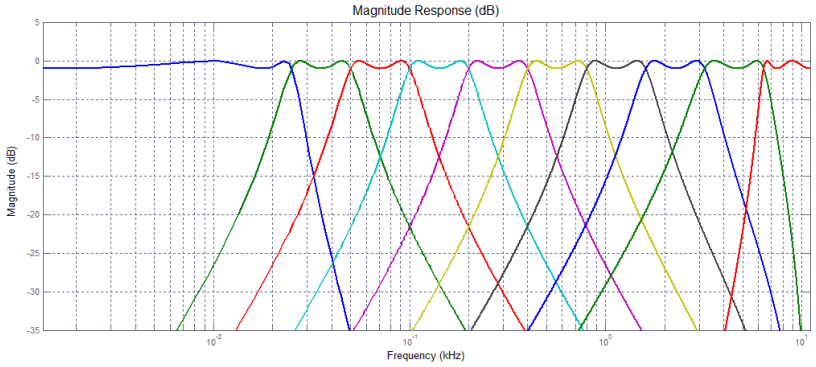


FIGURE 2.9: Frequency response of the 10-channel filter bank of octave scaled fourth-order elliptic filters.

For each channel the spectral flux was computed using Euclidean distance between successive magnitude spectra. The Euclidean distance is given by

$$D_E(p,q) = \sqrt{\sum_{i=1}^{N}(p(i)-q(i))^2} \qquad 2.6$$

where *p(i)* and *q(i)* are *N*-dimensional feature vectors. In our implementation, we apply DCT to the spectral flux to decorrelate the outputs. The outputs are then called spectral flux coefficients.

2.3.2.4 Spectral distribution descriptors

To enhance the timbre model, a number of spectral features are also derived to describe the distribution of the spectrum and may be correlated to the perceptual characteristics of the audio. These features are based on the magnitude spectrum and are calculated for every frame of sound. Spectral descriptors were used by Tzanetakis and Cook (2002) to describe *timbral texture*. The following features are also derived in our system:

1) *Spectral centroid* – defined as the center of gravity of the magnitude spectrum of the STFT

$$\mu_1 = \frac{\sum_{n=1}^{N} nM_t(n)}{\sum_{n=1}^{N} M_t(n)} \qquad 2.7$$

where $M_t(n)$ is the magnitude of the Fourier transform at frame t and frequency bin n. The spectral centroid is correlated with the perception of brightness of a sound (Grey 1978). Sounds with "dark" qualities tend to have more low frequency content whereas "bright sounds" have more high frequency content.

2) *Spectral spread* – describes the dispersion or spread of the magnitude spectrum.

$$\sigma^2 = \mu_2 = \frac{\sum_{n=1}^{N}(n-\mu_1)^2 M_t(n)}{\sum_{n=1}^{N} M_t(n)} \qquad 2.8$$

The value of the spectral spread may be proportional with the richness or fullness of a sound. For example, a violin sounds fuller than a piano. This is reflected on a violin's wide spectrum compared to a piano, see Schwarz (1998) for illustrations.

3) *Spectral skewness*– describes the symmetry of the magnitude spectrum.

$$\mu_3 = \sum_{n=1}^{N}\left(\frac{M_t(n)-\mu_1}{\sigma}\right)^3 \qquad 2.9$$

A symmetrical magnitude spectrum has a skewness value of 0. A positive value often indicates that the spectrum exhibits a concentration of mass toward the left and a long tail to the right whereas a negative value indicates the opposite.

4) *Spectral kurtosis*– measures whether the distribution of the magnitude spectrum is shaped like a Gaussian distribution or not.

$$\mu_4 = \sum_{n=1}^{N}\left(\frac{M_t(n)-\mu_1}{\sigma}\right)^4 \qquad 2.10$$

Similar to skewness, kurtosis is a descriptor of the shape of a magnitude spectrum.

5) *Spectral flatness* – indicates whether the magnitude spectrum is smooth or "spiky".

$$SF = \frac{\exp\left(1/N\sum_{n=1}^{N}\log\left(M_t(n)+10^{-20}\right)\right)}{1/N\sum_{n=1}^{N} M_t(n)} \qquad 2.11$$

The constant in the numerator prevents computing errors. The upper limit depends on the maximum value of the magnitude spectrum. Low values imply non-flat (e.g. tonal) spectrum whereas high values imply a flat (e.g. noisy) spectrum.

6) *Spectral flux* – defined as the squared difference between the normalized magnitudes of successive unfiltered spectral distributions. It measures the amount of local spectral change.

$$SFlx = \sum_{n=1}^{N} \left(M'_t(n) - M'_{t-1}(n)\right)^2 \qquad 2.12$$

where $M'_t(n)$ and $M'_{t-1}(n)$ are the normalized magnitude spectrum of the Fourier transform at the current frame t, and the previous frame, t-1, respectively. It can be interpreted as a basic approximation to the sensation *roughness* which describes a modulation in the excitation pattern levels (Fastl and Zwicker 2006).

7) *Spectral roll-off* – defined as the frequency R_t in Hertz below which 85% of the magnitude distribution is concentrated.

$$\sum_{n=1}^{R_t} M_t(n) = 0.85 \sum_{n=1}^{N} M_t(n) \qquad 2.13$$

The roll-off is a measure of spectral shape. High values indicate significant magnitude components at high frequencies; thus, a high audio bandwidth.

8) *Spectral brightness* – measures the amount of energy above the cut-off frequency of 1500 Hz. This is related to the sensation *sharpness* that describes the high frequency content of a sound (Fastl and Zwicker 2006).

$$SB = \frac{\sum_{n=n'}^{N} M_t(n)^2}{\sum_{n=1}^{N} M_t(n)^2} \qquad 2.14$$

where n' corresponds to the frequency bin that contains the frequency 1500 Hz.

9) *Spectral entropy* – indicates whether the magnitude spectrum has predominant peaks or not. The definition of entropy was introduced by Claude Shannon (Shannon 1948).The entropy is maximal when the spectrum is flat, corresponding to a situation of maximum frequency uncertainty. The spectral flatness, Equation 2.11, and spectral entropy are different mathematical representations of the noisiness of the spectrum.

$$SE = -\frac{\sum_{n=1}^{N} M'_t(n) \log\left(M'_t(n)\right)}{\log(N)} \qquad 2.15$$

The scalar values from the nine spectral descriptors are concatenated to form a feature vector as shown in FIGURE 2.10.

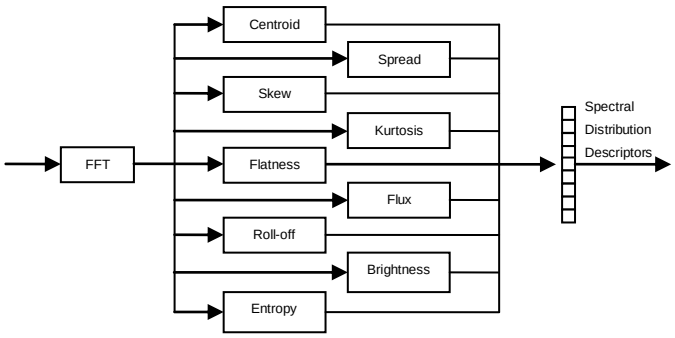


FIGURE 2.10: Block diagram of spectral profile descriptors.

2.3.3 Summarizing timbre features

The features derived from each audio track must be summarized efficiently and take into consideration the similarity computation method that will be performed. In this work, the features are computed for each time segment or *frame*. These features are aggregated for every song using the bag-of-frames approach to model global statistics. The bag-of-frames approach is more appropriate in this case since the target application is low specificity audio similarity estimation, refer to Figure 2.3. We enumerate some alternative models used in the literature before describing mathematical models of the bag-of-frames approach.

2.3.3.1 Vector quantization models

Vector quantization is a non-parametric distribution model. Initial studies that used frame-level features were based on tree-based vector quantization (VQ) (Foote 1997, Pye 2000). The training songs are parameterized into feature vectors. Each training example is associated with a particular music genre. A quantization tree is grown that partitions the feature space into regions that have maximally different class populations. All training feature vectors are passed through the tree into the leaf cells. The relative quantity of the samples in each cell forms a histogram template. The templates are compared with the histogram of a query song to provide an estimate of acoustic

similarity. The comparisons are computed by Euclidean distance or cosine similarity. The study by Pye (2000) compared the performance of vector quantization and Gaussian Mixture Models (GMM). The results showed that GMM outperformed the vector quantization scheme.

Standard k-means VQ is computationally expensive. To solve this, Seyerlehner et al. (2008) proposed a multi-level clustering architecture. They were able to reduce the number of feature vectors by randomly sub-sampling the overall distribution of feature vectors. Redundant feature vectors from a single song increase computational cost but do not improve the quality of the global partitioning of the feature space. Hence, they used the k-means++ (Arthur and Vassilvitskii 2007) algorithm to cluster the features within individual songs then pass the song-level cluster centres to the global codebook generation, where another k-means++ algorithm is used to generate the final codebook. They claimed that this approach is comparable to the state-of-the-art in terms of quality. Moreover, it does not suffer from the *hub* problem.

2.3.3.2 Hidden Markov Models

The bag-of-frames approach discards the sequential order of the frames. However, the study by Grey (1977) on perception of musical timbre of single musical instruments highlight the importance of the temporal aspect of audio signals. A popular method to incorporate the temporal aspect to music processing is the use of Hidden Markov Models (HMMs). The HMM is widely used in speech recognition, refer to Rabiner and Juang (1986) for a more detailed description of HMMs. Given a sequential data, the aim of HMM is to obtain a model that can be used later to identify or recognize other sequences of data. The HMM is composed of a set of Gaussian Mixture Models, known as states, that are linked with a transition matrix which indicates the probability of going from one state to another in a Markov process. A stochastic process has a Markov property if the future state depends solely on its present state just as well as knowing the full history of the process.

The study by Aucouturier and Pachet (2004) on timbre similarity showed that the performance of HMM is no better than static Gaussian Mixture Models. The authors suggested that the dynamics modelled by HMMs are not meaningful since the source is polyphonic, i.e. contains multiple instruments that are not synchronised. In addition, the process of comparing HMM is computationally expensive. It involves Monte Carlo sampling of the HMM to generate sequences then computing the log likelihood of each of these sequences given the other models. In another study by Flexer et al. (2005), the authors also compared HMM to GMM for describing spectral similarity of songs. Their results showed that HMM can better describe the spectral similarity of songs but does not result in any gain in genre classification accuracy. The authors claimed that this is the reason why HMM is not widely used for music analysis in literature.

2.3.3.3 Deep belief network

An alternative method for robust feature representations based on *deep learning* is becoming popular (Humphrey et al. 2012). For example, Deep Belief Networks (DBNs) have been applied to frame-level tasks such as instrument classification (Hamel et al. 2009), genre identification (Hamel and Eck 2010), and mood prediction (Schmidt et al. 2012). The aim of DBN is to learn more abstract representations of the input data in a layer-wise fashion using unsupervised learning. It is a neural network constructed from layers of Restricted Boltzmann Machines (RBM) (Bengio et al. 2007). There can be several layers of RBMs stacked on top of each other by linking the hidden layer of one RBM to the visible layer of the next RBM. Instead of using the computationally intensive gradient descent for the whole training of the neural network, it performs a greedy layer-wise unsupervised pre-training phase. The unsupervised pre-training phase produces the weights that are optimized later by supervised learning using gradient descent learning.

In Hamel and Eck (2010), the authors used DBN to *learn* feature representations from the magnitude spectrum of the audio. They used the learned features and MFCCs as inputs to SVMs to perform genre classification experiments. Results showed that the learned features outperformed MFCCs. Another benefit is that once the DBN is trained,

feature extraction can be performed quickly. However, the main disadvantage of this approach is the long computation time for training the DBN. Although deep learning has been successfully applied to classification tasks, it would be interesting to see how this approach can be applied to music similarity tasks.

2.3.3.4 Gaussian distribution

In the bag-of-frames approach, features can be assumed as independent and identically distributed data. In this case, temporal information is ignored. This assumption allows a probability distribution to model the features. Existing literature in audio music classification, such as Tzanetakis et al. (2001), model the spectral information with a single Gaussian distribution with a diagonal covariance matrix. Other works (Logan and Salomon 2001, Aucouturier and Pachet 2002, Aucouturier et al. 2005) use Gaussian Mixture Models to model the distributions using *k*-means and expectation-maximization algorithms. Subsequent works by Mandel and Ellis (2005) and Pampalk (2006b) have shown that the same level of performance can be achieved using a single Gaussian distribution with a full covariance matrix. In this work, the single Gaussian with full covariance approach is implemented to benefit from reduced computational complexity compared to Gaussian Mixture Models. A *d*-dimensional Gaussian probability density function is defined as:

$$N(x \mid \mu, \Sigma) = \left(\frac{1}{2\pi}\right)^{d/2} |\Sigma|^{-1/2} \exp\left(-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)\right) \quad 2.16$$

$$\mu = \left(\frac{1}{L}\right)\sum_{l=1}^{L} x_l \qquad \Sigma = \left(\frac{1}{2L}\sum_{n=1}^{L} x_l x_l^T\right) - \mu\mu^T \quad 2.17$$

where x is the observation (D-dimensional feature), L is the number of observations, μ is the mean, and Σ is a d x d covariance matrix. Thus, we only need to compute the mean and the covariance matrix from the feature vectors for each song.

Since the delta coefficients and the related spectral distribution descriptors are computed on the same time frames that are used for MFCC extraction, these coefficients may also be appended on the MFCC vectors before taking the mean and covariance. This increases

the dimensions of the mean vector and covariance matrix, thereby increasing computational complexity. Optimization is performed to find a balance between the number of dimensions and the performance of the algorithm. This is done by determining the best combination of MFCC values and other features that gives the best performance. For example, in each time frame the 20 MFCC values are appended with the 20 delta and 20 delta-delta coefficients. This results in a feature vector of length 60. The optimization process is discussed in more detail in Section 3.4.1.

2.3.4 Quantifying timbre similarity

To quantify timbre similarity, a distance measure or similarity function is applied between timbre models. There are a number of distance and similarity functions that have been used for audio similarity estimation. The key point in choosing the distance measure or similarity function is that it is appropriate for the model used to summarize the features.

2.3.4.1 Geometric distance measure

Assuming that the feature vectors exist as points in Euclidean space, the distance between two points is usually given by the Euclidean distance, see Equation 2.19. In general, the Minowski distance of order p, also known as p-norm distance, can be used:

$$1-norm = \sum_{i=1}^{N} | p_i - q_i | \quad 2.18$$

$$2-norm = \left(\sum_{i=1}^{N} | p_i - q_i |^2 \right)^{1/2} \quad 2.19$$

$$p-norm = \left(\sum_{i=1}^{N} | p_i - q_i |^p \right)^{1/p} \quad 2.20$$

where p_i and q_i are N-dimensional vectors. The 1-norm distance is also known as Manhattan distance whereas the 2-norm distance is the Euclidean distance.

Other distance functions that belong in this category include cosine similarity and Mahalanobis distance. The former is used to measure the angle between two vectors

whereas the latter uses the covariance matrix of the data to estimate the similarity between an unknown vector and a known sample set. They are defined as follows:

$$D_{cosine} = (p.q)/(|p||q|) \quad 2.21$$

$$D_{Mahalanobis} = \sqrt{(p-u)^T \Sigma^{-1} (p-u)} \quad 2.22$$

where p and q are vectors from the same distribution with mean u and covariance matrix Σ. If the covariance matrix is replaced by the identity matrix then the Mahalanobis distance becomes the Euclidean distance. The features used in this thesis are modelled with vectors and matrices as components. Hence, the geometric distance functions cannot be used directly.

2.3.4.2 Ground distance

The Earth Mover's Distance (EMD) was first used to compare the colour feature distributions generated from images (Rubner et al. 1998). EMD is based on the amount of work needed to change one distribution to another. It was applied as a music similarity function by (Logan and Salomon 2001). To do this, the feature vectors are clustered into groups which are similar using k-means algorithm. The features can then be represented by a signature comprising of k triples, each being the mean, covariance matrix and weight of a cluster. When comparing two signatures, one signature can be thought of as mounds of earth whereas the other is a set of holes. The distance is the minimum amount of work needed to move the earth into the holes. The work, in this example, is the mass moved multiplied by the distance it is moved. The minimum amount of work necessary is determined. The main disadvantage of this approach is that it is computationally expensive.

2.3.4.3 Information theory similarity function

The Kullback-Leibler divergence is also known as *relative entropy* or *information gain* (Kullback and Leibler 1951). The divergence computes the difference between two distributions p and q:

$$KL(p|q) = \int p(x) \log \frac{p(x)}{q(x)} dx \quad 2.23$$

The divergence is non-symmetric and non-negative. The triangle inequality also does not hold, hence the divergence is not a metric distance function. The timbre similarity between two tracks can be computed using the closed form solution of the Kullback-Leibler (KL) divergence for single multivariate Gaussian distributions. The KL divergence between two single Gaussians $p(x)=N(\mathrm{x};\mu_p,\Sigma_p)$ and $q(x)=N(\mathrm{x};\mu_q,\Sigma_q)$ of the dimension d is given by (Penny 2001):

$$KL(p\,|\,q)=\frac{1}{2}\left(Tr\left(\Sigma_q^{-1}\Sigma_p\right)+\left(\mu_q-\mu_p\right)^T\Sigma_q^{-1}\left(\mu_q-\mu_p\right)-\log_e\frac{|\Sigma_p|}{|\Sigma_q|}-d\right) \qquad 2.24$$

where $|\Sigma|$ denotes the determinant of the matrix Σ, $Tr(M)$ denotes the trace of the matrix M. It is crucial that the covariance matrix Σ is nonsingular. Otherwise, this equation will result in an error since the inverse of the matrix does not exist. Computational error may also arise if the audio clip has silent or very soft parts as the inverse of the matrix may contain very large values. These conditions are managed during audio pre-processing in our system before the actual feature extraction. First, the audio signal is normalized to boost the overall audio level. Second, in the rare cases wherein the audio signal is silent, a random signal with the same length as the original is added. Third, audio signals that are less than the threshold of hearing, i.e. around 3 dB, are replaced with random values whose sound pressure level is around 3 dB (Fastl and Zwicker 2006).

Since the KL divergence is not symmetric, the equation can be symmetrised by getting the sum in both directions:

$$SKL(p\,|\,q)=\frac{1}{2}\left(KL(p\,|\,q)+KL(q\,|\,p)\right) \qquad 2.25$$

Several researchers have proposed methods to make SKL behave like a metric distance function. The similarity estimation may be objectively evaluated on how often the triangular inequality is satisfied. The extent to which the triangular inequality does not hold can be measured by computing the percentage of all triangles in a dataset that do

not obey the inequality. As this may be time-consuming for large databases, random sampling can be used. Pampalk (Pampalk 2006b) proposed rescaling the SKL as:

$$\delta(p,q) = 1 - e^{\lambda SKL(p|q)} \quad 2.26$$

Using this approach, the inequality holds on average for 29% of the cases for single Gaussian algorithm, and 36% for single Gaussian combined with simple features. In the 2006 submission to MIREX, Pampalk corrected this by setting the contribution of spectral similarity to zero whenever numerical problems occurred when computing the inverse covariance matrix (Pampalk 2006a). Schnitzer et al. (2009) proposed an alternative to rescale SKL:

$$\delta(p \mid q) = \sqrt{SKL(p \mid q)} \quad 2.27$$

Results of experiments on a dataset of 100,000 randomly drawn Gaussian timbre models are listed in Table 2.1. This shows that the square root function performs better than the exponentiation approach. Finally, Charbuillet et al. (2010) proposed and evaluated on a large-scale database a transformation function that turns SKL divergence into an exact metric.

$$\delta(p \mid q) = \sqrt{\log\left(SKL(p \mid q) + 1\right)} \quad 2.28$$

They also reported that this function preserves the similarity ordering. Hence, we also use this transformation function.

TABLE 2.1: Percentage of Gaussian object triples fulfilling the triangle inequality(Schnitzer, Flexer, and Widmer 2009)

Divergence	*% triangle inequality held*
$SKL()$	91.57%
$1-e^{\lambda SKL()}, \lambda = -\frac{1}{100}$	93.71%
$1-e^{\lambda SKL()}, \lambda = -\frac{1}{50}$	95.60%
$\sqrt{SKL()}$	99.32%

2.4 Monophonic versus polyphonic timbre

Monophonic timbre refers to the sound of a single instrument played at a specific note whereas polyphonic timbre refers to the global mixture of several instruments. In this section, the correlation between the derived timbre features and the actual perception of timbre is described.

2.4.1 Monophonic timbre

Perceptual research had been done to determine the possible acoustic attributes that best correlate with monophonic timbre. Monophonic timbre is the texture of the sound produced by an instrument playing a single note. Early studies conducted by Grey (1975), Grey (1977) and Wessel (1975) employed the verbally simple notion of 'similarity rating' to build a timbre space. The numerical values of similarity scores are collected for a set of instrumental notes played at a specific pitch. Successive studies by Iverson (1993) and McAdams et al. (1995) used synthetic sounds to create hybrids between instruments. Multidimensional scaling (MDS) (Kruskal 1964, Shepard 1962b, Shepard 1962a) is then used to map similarity ratings into a low-dimensional geometric space where distances in the space correspond to perceived similarity.

The physical timbre space may have two or three dimensions where each dimension corresponds to a perceptual feature. Figure 2.11 shows an example of a timbre space by McAdams et al. (1995). The major dimensions used in this feature space are spectral centroid, rise time, and spectral flux. The spectral centroid, or the centre of mass of a spectrum, corresponds to the perception of "brightness" of a sound. The rise time is the amount of time between the onset of the sound and the maximum amplitude (Iverson 1993). The onset is the beginning of discrete events in acoustic signals. The spectral flux describes the change in energy in the audio signal. The points in the spatial model represent the 18 instruments used in the study, e.g. harp (hrp), piano (pno), etc.

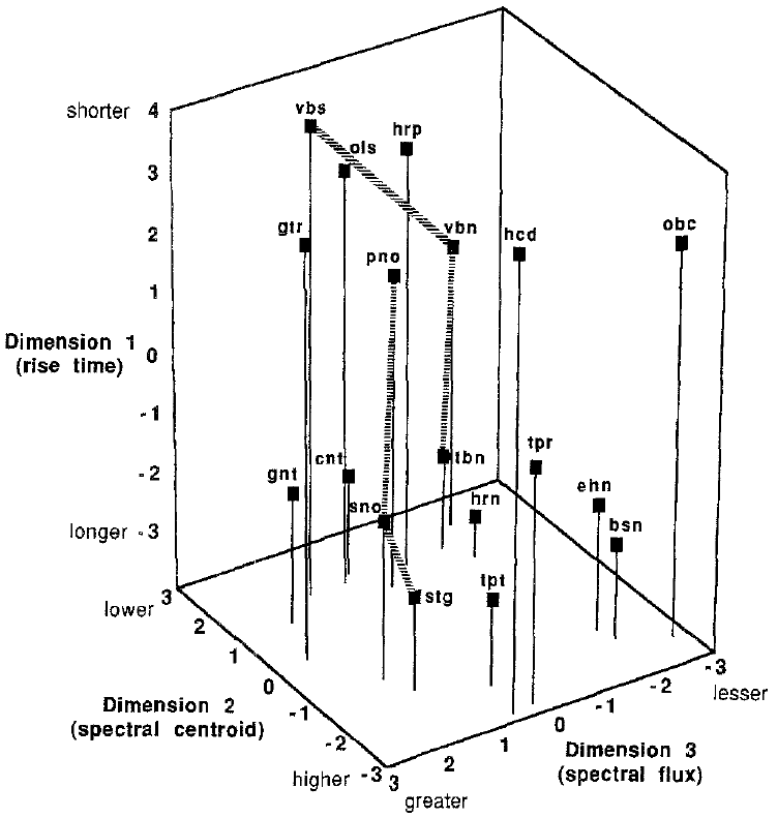


FIGURE 2.11: Timbre space in 3-*d* derived from dissimilarity ratings on 18 timbres by 88 subjects. The acoustic correlates of the perceptual dimensions are indicated in parentheses. Reproduced from (McAdams et al. 1995)

The studies by Grey (1977), Wessel (1975), Iverson (1993), and McAdams et al. (1995) agree that the spectral centroid, defined in Equation 2.7, is a major perceptual dimension. The second dimension corresponds to the attack portion or rise time of the sound (McAdams et al. 1995). The nature of the third dimension varies among researchers: spectral variation over time (Grey 1977), "spectral fine structure" (Krimphoff et al. 1994), spectral flux (McAdams et al. 1995).

The studies by De Poli and Prandoni (1997) and Terasawa et al. (2005) proved that MFCCs are well suited to create a timbre space. The clustering of instruments in the timbre space are similar to previous MDS studies (De Poli and Prandoni 1997). Furthermore, a model for timbre based on MFCCs accounts for 66% of the perceptual variance in human timbre similarity judgments for steady-state, individual sounds (Terasawa et al. 2005).

2.4.2 Polyphonic timbre

Most of the studies described in the preceding section focused on sound samples corresponding to clean recordings of single instruments, played on a specific note. This is unrealistic for possible music applications that usually have multiple instruments playing simultaneously. The overall timbral mixture in a music signal is called polyphonic timbre. It is an agglomerate of spectral and rapid time-domain variability in an acoustic signal formed in a manner comparable to the Gestalt effect that enables listeners to identify, classify, and categorize the heard piece of music (Gjerdingen and Perrott 2008). For example, the presence of high amounts of acoustic energy in the lower end of the spectrum leads towards rap and hip-hop rather than classical music (Gjerdingen and Perrott 2008).

Polyphonic timbre has been proven to be an important perceptual component of music, especially in studies that involve tasks such as genre identification, categorization, or emotional affect attribution. The psychological study performed by Gjerdingen and Perrott (2008) demonstrated that it can take as little as 250 ms for people to identify or classify music excerpts into genres. This implies that rapid genre identification does not require other features such as melody, harmony, and rhythm. They suggested that listeners can achieve a global categorization of genre at least as fast as they can categorize component features. It has also been shown that listeners can recognize the affective connotations of sad or happy musical excerpts taken from Western classical music, even when they are as short as half a second. This is most likely due to the overall timbral and spectrotemporal properties of the excerpt (Peretz 1998). Timbre features provide the best performance in music emotion recognition systems when used as individual features (Barthet et al. 2013).

2.5 Polyphonic timbre in music computing

Features representing polyphonic timbre have been found essential in the design of computational systems for audio similarity estimation and categorization according to genre, emotions, etc. For example, many studies have focused on timbre and rhythm

features when designing such computational systems. The following paragraphs describe the algorithms that have provided the groundwork for audio similarity estimation. The performance of the algorithms are usually characterised in terms of genre classification accuracy. Genres exist as a group of stylistic tendencies, codes, conventions, and expectations that become meaningful in relation to one another at a particular moment in time (Hesmondhalgh and Negus 2002). Hence, music from similar genres may sound similar assuming that they have the same instrumentation and follows the styles and conventions of that particular genre.

2.5.1 Logan and Salomon

Logan and Salomon were one of the first to publish a music similarity function based on audio content analysis (Logan and Salomon 2001). The similarity function has been successfully applied to playlist generation, artist identification and genre classification of music. The method is based on the comparison of a 'signature' for each track using the Earth Mover's Distance (EMD), a mathematical measure of the difference between two distributions. For each track, the MFCCs are computed. The signature is then formed using k-means clustering on the computed MFCCs (Macqueen 1967).

Three evaluations were presented. First, they determined the optimum number of MFCCs that form the signatures. Second, they used listening tests to compare their best parameter settings against random shuffling. Third, they measured how well the original version of the song can be retrieved given a clipped version of the original as query. A clipped version is a song where a section of random length of up to 30 s from a randomly selected place in the song is removed.

Logan and Salomon report the average number of similar songs (in the top 5, 10, and 20 ranked positions) compared to random and show that their approach works significantly better. For example, the random scheme finds in average 0.9 similar songs in the top 20 whereas their approach finds 8.2 songs.

2.5.2 Aucouturier and Pachet

Aucouturier and Pachet used a timbral similarity measure based on the Gaussian model of cepstral coefficients (Aucouturier and Pachet 2002). A Gaussian Mixture Model (GMM) is derived from MFCC vectors from each song and is compared by sampling the distributions to estimate the timbral similarity between two songs. Since the original MFCC vectors are not available from this point, Monte Carlo sampling is used to generate samples for the likelihood computation. The authors improved the model by optimizing parameters such as the number of clusters and MFCCs. They found that a mixture of three Gaussian distributions was sufficient to model the MFCC distribution of most songs. For the MFCCs, they measured the optimum dimension to be around 10 coefficients.

They reported the following information: (1) the average number of songs in the same genre from a database of 17,075 music titles (closest 1, 5, 10, 20 and 100 songs), and (2) the measured "overlap on same genre" and "overlap on different genre." The former describes the proportion of songs which have different genre as the query, but whose distance to the query is smaller than the mean distance to songs of the same genre. The latter describes the proportion of songs which have the same genre as the query, but whose distance to the query is larger than the mean distance to songs of different genre. Their results showed that there is a very poor correlation between genre and timbre. The precision of a query on genre based on timbral distance is very low (14.1%). However, values for "overlap on same genre" (57.1%) and "overlap on different genre" (27.1%) are high.

Aucouturier and Pachet report that their system identifies surprising associations between certain songs, often from very different genres of music, which they exploit in the calculation of what they term an 'Aha' factor. 'Aha' is calculated by comparing the content-based 'timbral' distance measure to a metric based on textual metadata. Pairs of tracks identified as having similar timbres, but whose metadata does not indicate that they might be similar, are assigned high values of the 'Aha' factor. To correct this, they defined a weighted combination of their similarity metric with a metric based on textual

metadata. However, this contradicts the benefits of a purely content-based similarity system.

2.5.3 Mandel and Ellis

In 2005, Mandel and Ellis presented a new system that uses support vector machines to classify songs based on features calculated over their entire lengths (Mandel and Ellis 2005). Similar to (Aucouturier and Pachet 2002), the first 20 MFCCs are calculated for a given song. The mean and covariance matrix are computed for the resulting MFCC vectors. Thus, a song is represented by a 20 x 20 covariance matrix and a 20-dimensional mean vector. This is a single Gaussian distribution model. Similar to other timbre music similarity models, the temporal aspects are ignored. The similarity between two songs is then computed by Kullback-Leibler (KL) divergence (Kullback and Leibler 1951). Since the KL divergence is not symmetric, its symmetrised form is used.

Four setups were used in their experiments. (1) Neither song level features nor Support Vector Machines (SVM) were used, training a single GMM on the MFCC frames from all of an artist's songs at once. The likelihood of each song's frames was evaluated under each artist model and a song was predicted to come from the model with the maximum likelihood of generating its frames. (2) Support vector machines but not song-level features were used. By training an 18-way Directed Acyclic Graph SVM (DAG SVM) (Platt et al. 2000) on a subset of frames used in the first experiment, they attempted to learn to classify MFCC frames by artist. To classify a song, the frames are classified and the most frequently predicted frame class determined the song's class. (3) The third experiment used song-level features, but a simple k-nearest neighbours (kNN) classifier. From the song-level features, a kNN classifier was used to label test songs with the label most prevalent among the k nearest training songs. (4) The final setup used song-level features and an SVM classifier. For all song-level features, an 18-way DAG-SVM classifier for artists was learned.

Their results showed that song-level features and SVM classifiers had an advantage with a 15 percent point gain in 18-way classification accuracy. It was also highlighted that the

training times for the classifiers using low-level features were considerably higher than those using song-level features. The trade-off in the fast training times is the long feature extraction and computing distances between songs. An important observation is the "album effect" in which almost every classifier performs significantly better when trained and tested on songs from the same albums.

2.5.4 Pampalk

Elias Pampalk, Andreas Rauber and Dieter Merkl presented a music genre classification system based on psychoacoustic models (Pampalk et al. 2002). Their two-stage feature extraction process calculates a time-invariant representation for each song called *rhythm pattern*. The rhythm pattern contains information on how strong and fast beats are played within the respective frequency bands. Similarity is then computed simply as the Euclidean distance between two rhythm patterns. Unlike other approaches, their system considers temporal information but disregards vital timbre information. Although the results obtained are generally satisfactory, unexpected clustering of songs highlight the limitations of their approach.

To overcome this, Elias Pampalk combined a spectral similarity model with information from fluctuation patterns (FP) (Pampalk 2006b). The spectral similarity model is based on the work of Mandel & Ellis (Mandel and Ellis 2005). The fluctuation pattern is derived from the rhythm pattern in Pampalk et al. (2002). In addition, the "gravity" and "bass" descriptors are extracted from the fluctuation pattern.

The spectral similarity model is derived by computing the MFCCs for a 30-second audio clip from the center of the piece. In particular, a 19-dimensional MFCC vector for every 23 ms of the signal is computed. The distribution of these vectors is summarised using a single Gaussian with full covariance matrix. The distance between two Gaussians is computed using a symmetric version of the Kullback-Leibler divergence.

The fluctuation pattern (FP) describes the modulation of the loudness amplitudes per frequency bands. To some extent it can describe periodic beats. The resulting fluctuation

pattern is a matrix with rows corresponding to frequency bands and columns corresponding to modulation frequencies (in the range of 0 to 10 Hz). The FPs are then summarized by computing the median of all FPs. The distance between pieces is computed by interpreting the FP matrix as high-dimensional vector and computing the Euclidean distance.

There are several features that can be extracted from the fluctuation patterns. However, it was found that FP "gravity" and FP "bass" are significant in audio similarity estimations. The "gravity" describes the center of gravity of the FP on the modulation frequency axis. Its values indicate that the piece might be perceived as slow or fast. However, gravity is not intended to model the perception of tempo. Effects such as vibrato or tremolo are also reflected in the FP. The "bass" is calculated as the sum of the values in the two lowest frequency bands with a modulation frequency band higher than 1 Hz. The distance of two songs for each of these descriptors is computed as the absolute difference of values.

Given the four distance values, the overall similarity of two pieces is computed as a weighted linear combination. The normalization and weights used are described in (Pampalk 2006b). This system was submitted to the MIREX'06 (Music Information Retrieval eXchange) evaluation track on audio-based music similarity and retrieval. Overall, this implementation performed slightly better than the other implementations. However, the measured differences were insignificant (Pampalk 2006a).

2.5.5 Flexer, Pampalk and Widmer

In 2005, Flexer et al. compared Hidden Markov Models (HMMs) to Gaussian Mixture Models (GMMs) for describing spectral similarity of songs (Flexer et al. 2005). Direct comparisons were based on the log-likelihood of songs given an HMM or GMM. Although HMMs seem to better describe spectral similarity of songs, there was no significant gain in terms of genre classification accuracy.

Unlike GMMs, HMMs incorporate temporal context by statistically modelling the locally stationary data and their transition probabilities. To create a model for a song, the first eight MFCCs are computed. The HMMs are then trained with the MFCC values using a Gaussian Observation Hidden Markov Model (GOHMM). The Expectation-Maximization (EM) algorithm is used to train the GOHMM. The forward algorithm is used to identify most likely state sequences corresponding to a particular time series and enable the computation of log-likelihoods.

Their results showed that there is no significant difference in genre classification performance between any of the model types. They inferred that this may be the reason why there is little success in using HMM for music similarity as reported in the literature.

2.6 Research challenges

This section presents two of the major research and experimental challenges for this study.

2.6.1 Efficient similarity estimation

At present the largest digital music store, iTunes, contains over 37 million tracks[26]. Despite technological advances, content-based MIR is not integrated with iTunes. Next generation MIR systems must address this and bring content-based methods to the larger music services and digital libraries. Consider the problem of music recommendation to users. The recommendation system must *know* about as much music as possible to make good recommendations. Manual discovery approaches do not scale well given the rate of new music produced. On the other hand, there are certain processes that content-based approaches are very good and fast at doing with music, e.g. determining the key, tempo, or loudness. The acoustic information in combination with automatically mined textual

[26] http://www.apple.com/itunes/what-is/

information of music related data can create a scalable and practical solution for music recommendation systems.

As an illustration, the MIREX audio similarity experiment requires computation of a 7000 x 7000 similarity matrix, so the task required pair-wise track comparisons on the order of millions. However, today's music download services are on the order of tens of millions of tracks, so pair-wise comparisons require computation on the order of hundreds of trillions. Depending on the complexity of the similarity estimation algorithm, pair-wise methods may be inefficient at this scale on even the most advanced hardware. However, this scale is required to make content-based MIR suitable for solving today's media search and retrieval problems.

2.6.2 Description of polyphonic timbre

To resolve the vagueness in defining audio similarity, this work will focus on timbre similarity. This means searching for music with similar sound textures. The main objective is to determine a quantitative model of timbre of a musical signal. The task is straightforward for monophonic music where the single instrument playing can be modelled precisely using its spectral envelope. With polyphonic music, the usual approach is to treat the signal as a whole such that the timbre model represents a mixture of multiple instruments playing simultaneously.

A method that can improve the description of polyphonic timbre is source separation to extract information about the individual instruments in the signal. This is one of the most challenging tasks in music processing. Despite advances in source separation methods, the current techniques cannot handle arbitrary number of instruments playing simultaneously (Benetos et al. 2013). Thus, we have to review and improve the different models for polyphonic timbre.

2.7 Summary

This chapter provided an overview of approaches and techniques related to music information retrieval. Different approaches for audio similarity estimation were presented with focus on content-based techniques. The advantages of content-based audio retrieval were enumerated. Clearly, this approach is suitable for searching from untagged collections of music. It is recognized, however, that content-based methods serve to complement searches using tags and metadata.

The basic components of a content-based retrieval system were described. The crucial process in the system is feature extraction. Some of the common audio features and feature extraction algorithms used in the literature were described. These are usually low-level features derived from short time segments. A semantic gap exists between these low-level features and high level music concepts. It would seem that closing this gap goes beyond using content-based methods only, or at least using only low-level features. Since this research is limited to using low-level features and similarities between feature spaces, it is unable to directly close the semantic gap.

Different ways of summarizing or modelling the features were enumerated. The literature suggests that the bag-of-frames framework is practical for our target application. The single multivariate Gaussian distribution was selected to model the features for several reasons: 1) it has been proven to perform as well as a mixture of Gaussian distributions; 2) it is computationally efficient since it only requires the mean and the covariance matrix of the feature vectors; and 3) a closed form solution of the Kullback-Leibler divergence exists for single multivariate Gaussian distributions that can be used for similarity estimation.

The similar approaches by authors on timbre modelling highlight practical views on music computing. This means that the features and models derived do not accurately represent music cognition. But, the systems have enough intelligence to predict high-level descriptions of music signals, e.g. genre. The simple reason for this is that the perceptual aspects of polyphonic timbre have not been studied extensively. Hence, most of the

approaches are direct extensions of works on monophonic timbre. The signal is cut into short overlapping frames, and then a feature vector is computed for each frame. The most common features used to represent the overall timbre of music signals are the MFCCs. All feature vectors are summarized as a global distribution of the features for a particular music track or class, e.g. classical or rock in the case of genre classification system. A similarity measure that is appropriate for the global distribution model is used to compute similarity between tracks. The algorithms for feature extraction and modelling are implemented in Matlab® R2009b, see Appendix 2 for the scripts.

This study aims to build a better computational model for timbre by looking back at the major attributes of timbre. The successful use of MFCCs suggests that they capture some important properties of the overall timbre, specifically the spectral envelope. In addition, we will include two other attributes: 1) the range between tonal and noise-like character, and 2) the temporal unfolding and shaping of sound spectra. The two other attributes of timbre, time envelope and onset of a sound, are based on the time-domain waveform but due to the complex nature of polyphonic timbre, these attributes are hard to model. This requires separation of individual instruments, which is already beyond the scope of this research.

Chapter 3: **Estimating Timbre Similarity**

This chapter begins with a discussion on the representation of digital audio signals, which is the first step in timbre similarity estimation. This is followed by describing the design parameters that are considered to optimize the similarity estimates.

In this chapter, we also evaluate the different features described in Chapter 2 in the context of timbre similarity. In particular, we test several hypotheses that will lead to improvements in our system: 1) the performance of an algorithm can be improved by tuning the parameters of the features, 2) different features have their corresponding advantages, and 3) combining different timbre features will improve the overall performance of the system.

An ideal audio retrieval system should make the same judgment of similarity as the average human listener. Thus, it is common for researchers to validate the performance of their system with human subjective testing. This approach is necessary but usually involves a lot of time and resources to build a credible evaluation. In addition, it is difficult to create a standard scale for humans to base their judgement since each individual has his own notion of similarity. In this regard, objective testing is more commonly used since standard performance metrics are derived from distance matrices produced by the retrieval system, which takes fewer amount of resources than subjective testing.

One way to evaluate timbre similarity is through music genre similarity. The basic assumption is that timbrally similar songs most probably belong to the same genre. In a study by Gjerdingen and Perrott (2008), it was found that humans can perform genre classification in as short as 250 ms. It was argued that timbre encompasses all the spectral and rapid time-domain variability in the acoustic signal. Such information can be highly indicative of particular genres. Other features, such as melody or rhythm cannot be derived from such short audio clips. Given the distance measures between a query song and songs in a database, a finite number of candidate songs are returned. The

returned list is evaluated by determining the proportion of songs that has the same genre as the query song.

The choice of performance measures will affect interpretation of results. Ideally, we want the measures to reflect real world user queries. We enumerate the common performance measures used in information retrieval and discuss their limitations. We apply the performance measures in evaluating the individual feature spaces. We then propose a framework of combining feature spaces to improve the overall performance as well as to reduce the presence of hub and orphan songs.

3.1 Representing audio signals

An audio signal is a representation of sound as a function of time. It has a frequency range of around 20 to 20,000 Hz (the limits of human hearing). In the real world, it exists as an analogue signal. To perform digital signal processing, the analogue signal is transformed into a digital signal using an analogue-to-digital converter (ADC). An ADC samples the analogue signal at a fixed time interval, called sampling period and quantizes each amplitude sample by approximating the continuous amplitude values by a set of discrete values. The reciprocal of the sampling period is called the sampling rate or sampling frequency. Let the analogue signal be denoted as $x_a(t)$. If the analogue signal is sampled at a rate of F_s, the output is a discrete-time signal $x_a(n/F_s) \equiv x(n)$. For quantization, the number of possible discrete values is determined by the resolution, usually expressed in bits.

For example, a CD-quality audio signal uses a sampling rate of 44.1 kHz and the sampled amplitudes are encoded as 16-bit Pulse-Code Modulation (PCM) symbols. This means that there are 44100 samples for every second with 2^{16}=65536 possible values. This *uncompressed* format can easily be stored in physical media but takes up a considerable amount of space, e.g. a three-minute song on a CD is 30-40 megabytes in size.

Digital audio files are usually available for download in a *compressed* format to save storage space, e.g. a song that takes up 30 MB on a CD takes up only 3 MB in MP3

format. For digital signal processing, it is necessary to decode and resample the compressed audio files. Decoding reconstructs the raw digital audio signal in PCM format. Resampling ensures that all input signals for feature extraction have the same sampling rate. A lower sampling rate such as 11 kHz or 22 kHz is commonly used to reduce the amount of data to be processed. This process is called downsampling or subsampling.

Based on the sampling theorem, a band-limited continuous-time signal with bandwidth B, can be recovered from its samples provided that the sampling rate is at least $2B$ (Proakis and Manolakis 2006). If the sampling rate is lower than the original bandwidth of the signal, *aliasing* occurs. The sampling of a signal with a sampling rate F_s results on a periodic repetition of a signal's magnitude spectrum with period F_s. If $F_s<2B$, the shifted replicas of the magnitude spectrum overlap. The corresponding spectrum is obtained by adding the shifted portions with respect to the folding frequency $F_s/2$. Hence, the corresponding magnitude spectrum of a subsampled signal contains frequency components that do not exist in the original.

Audio pre-processing

The first step before calculating any features is to normalize the audio signal. The process scales and shifts the sound vectors so they have maximum amplitude of one and have an average value of zero,

$$x_{norm}(n) = \frac{x(n) - \bar{x}}{\max(|\, x(n) - \bar{x}\,|)} \qquad 3.1$$

where $x(n)$ is the sound vector and, $\bar{x}$ is the mean of the signal. This removes DC component from the Fourier transform and also ensures that the amplitude of the transforms are of similar magnitude. The DC component is simply the average value of the signal. This also prevents any problem that may result in computing matrix inversion for spectral similarity estimation. Since the signals are almost of similar magnitudes after normalization, the original *loudness* information is lost.

After normalization, the audio signal is segmented into a sequence of frames of length 512 samples (~23 ms for a sampling rate of 22 kHz) with no overlap, see Figure 3.6. The length of the frame assumes the signal is pseudo-stationary, or at least its statistics are varying very slowly. A Hanning window, given by equation 3.2, is applied to each frame using equation 3.3.

$$w(n) = 0.5\left(1 - \cos\left(\frac{2\pi n}{N-1}\right)\right) \tag{3.2}$$

$$x_{seg}'(n) = x_{seg}(n) \times w(n) \tag{3.3}$$

where $w(n)$ is the weight applied to the segmented signal, N is the length of the segment, $x_{seg}(n)$ represents the segmented signal and $x_{seg}'(n)$ is the windowed signal.

The process of producing a segmented signal is equivalent to applying a rectangular window to the signal. The discontinuities in the segmented signal turn up as high frequencies in the spectrum, frequencies that were not present in the original signal. This appears as if the energy has leaked out into all other frequencies. This phenomenon is known as spectral leakage. This undesirable effect can be minimized by using window functions, e.g. Hanning window. The advantage of using a Hanning window is very low aliasing while its disadvantage is widening of the main lobe that leads to decreased resolution.

Figure 3.1 demonstrates the effects of windowing on a 1 kHz sinusoid. The top row shows the time domain signal and its corresponding magnitude spectrum. The magnitude spectrum shows a single peak at 1 kHz. The middle row shows the signal after applying a 23 ms rectangular window. Its corresponding magnitude spectrum exhibits a main lobe that contains most of the energy and side lobes. The side lobes contain energy in frequencies that should not be present. The bottom row shows the signal after applying a 23 ms Hanning window. We can see from its magnitude spectrum that the energy in the side lobes are significantly minimized, thus minimizing spectral leakage.

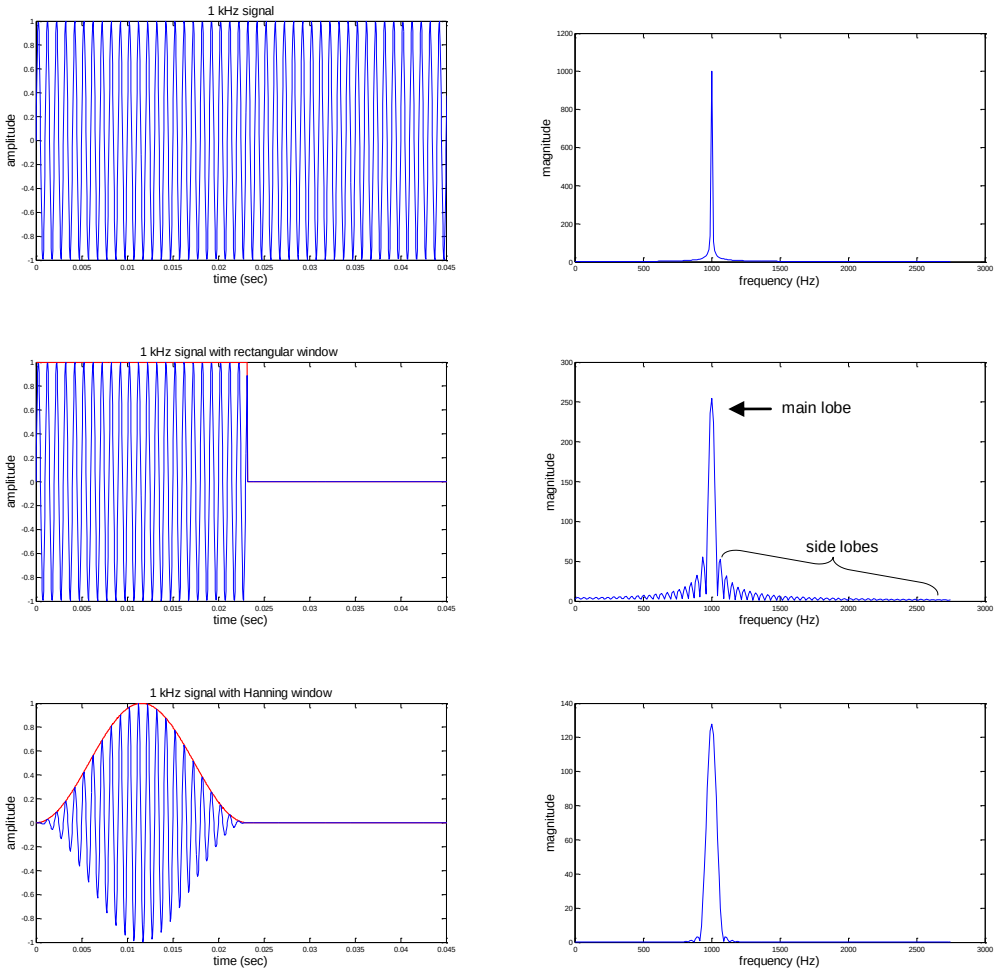


FIGURE 3.1: Effect of window functions on the magnitude response of a 1 kHz signal. The plots on the first column are time domain signals; the plots on the second column are their corresponding magnitude spectra.

The magnitude spectrum of the segmented signal is computed using the N-point Discrete Fourier transform (DFT) equation:

$$X(k) = \sum_{n=1}^{N} f'(n) e^{-2\pi j(k-1)(n-1)/N} \qquad k = 1, 2, \ldots, N \tag{3.4}$$

where $X(k)$ represents the k-th frequency magnitude of the current frame and N is the window size. Since $X(k)$ is even symmetric, only the first half is kept. An efficient method for computing DFT is Fast Fourier Transform (FFT), which can be performed by the Cooley-Tukey algorithm (Cooley and Tukey 1965). Based on Equation 3.4, the spectral

resolution is determined by the window size. Increasing the window size improves the spectral resolution.

3.2 Design parameters for optimization

An important aspect in designing any system is to optimize the parameters to achieve a desired performance. The following sections discuss the elements that are considered in the system's design.

3.2.1 Feature parameters

The first objective is to find the best feature parameter values. A *feature* or *descriptor* is a mathematical derivation of the input signal that aims to reduce its dimensionality and variability. For example, a 1 MB audio clip can be represented by a few kilobytes of features. The input files are usually normalized so that each signal will have the same range. Effective features should capture salient relevant information. In the case of modelling polyphonic timbre, we are focusing on the attributes of timbre as described from different psychoacoustic studies. Thus most of the features are derived from the magnitude spectrum since its properties affect the perception of timbre.

A typical parameter in feature extraction is the number of dimensions to be used, which affects the *feature space* where the model is derived. For example, the number of MFCCs extracted from each frame defines the dimensionality of the feature vectors. Increasing the dimension usually increases the ability to represent a particular feature, resulting in a better way to discriminate between classes of data.

Figure 3.2 illustrates a possible 2-class classification problem for a new data point denoted by a cross, using 2 arbitrary one-dimensional variables X and Y. Data points of the two classes are represented by dark gray triangles and light circles respectively. Using only 1 feature, neither X nor Y, is a good representation of the problem. For example, the observation of the value of $X \in [a,b]$ gives no clear indication about its most likely class: 5 items for each class are observed within its range. Similarly, the observation of

$Y \in [c,d]$ is ambiguous with respect to class. Now when both variables are jointly considered, $(X,Y) \in [a,b]\text{x}[c,d]$, then the most likely class in that region is "triangle" since there are three triangles and only two circles in that region. This suggests that increasing the number of features or dimensions usually improves modelling precision. However, it is often the case that increasing the number of features or dimensions can actually lead to degradation on modelling precision. This phenomenon is sometimes called the *curse of dimensionality* (Bellman 1961).

This problem can be understood using the same Figure 3.2. Suppose we add a third arbitrary dimension Z. If we keep data points constant and use the same regular spacing, then there will be 64 regions. This results in a more precise region in space. We can see that the number of regions grows exponentially with the dimensionality of space. The problem is that this will also require an exponential quantity of training data to ensure that the regions are not empty. In Figure 3.2, where the dimensionality is 2, there are 5 items found $[a,b]\text{x}[c,d]$. If we add a third dimension, it is possible that some of the regions are empty. In this case, the precision of the feature model fails.

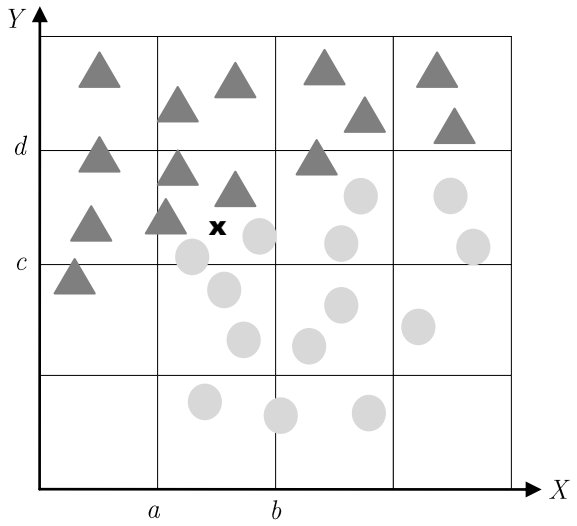


FIGURE 3.2: Illustration of the influence of feature dimension on a 2-class classification problem using two arbitrary one-dimensional variables X and Y.

Examples:

In Logan and Salomon (2001), the authors evaluated the influence of the number of MFCCs on the performance of a music similarity algorithm. Table 3.1 shows one of their results. It tabulates the average number of songs with the same genre as the query song. They observed a slight improvement in the performance as the dimensionality increases from 12 to 19. However, the performance degrades as the dimensionality increases to 29. This highlights the effect of the curse of dimensionality. Another undesirable effect of high dimensionality is the existence of *hub* songs (songs which frequently appear in the nearest neighbour list of other songs). In Berenzweig et al. (2003), the authors developed a conceptual framework that related the existence of hubs to the high-dimensionality of the feature space.

Table 3.1: Average number of closest songs with the same genre as the seed song (Logan and Salomon 2001)

Number of MFCC Features	*Average number of songs in the same genre*		
	Closest 5	Closest 10	Closest 20
12	3.43	6.53	12.4
19	3.44	6.57	12.5
29	3.36	6.44	12.3

3.2.2 Model parameters

The second objective is to find the best statistical model parameter values. Similar to the previous section, the statistical models used to learn the distribution of feature vectors have a number of fixed parameters that must be selected before training. These parameters affect the complexity of the model and its ability to properly model feature vectors. For example, the number of components in a Gaussian mixture model defines its flexibility or adaptability. Theoretically, the Gaussian mixture model is universal in that any source probability density can be approximated by a sufficient number of mixtures (Bishop 2007). However, increasing the complexity of the model does not necessarily improve the "fit" of that model to the data.

This can be explained in the case of a simple polynomial curve fitting problem shown in Figure 3.3. Suppose 10 training data are generated from the function $\sin(2\pi x)$ with some additional random noise. The goal is to model the data with a polynomial function of the form

$$y(x, w) = \sum_{i=0}^{M} w_i x^i \qquad 3.5$$

where M is the order of the polynomial. Figure 3.3 shows the plots of polynomials having various orders $M \in \{0,1,3,9\}$. The green curve shows the ideal function $\sin(2\pi x)$ while the red curves are the fitted polynomials. Models with $M=0$ and $M=1$ give poor fits to the data and consequently poor representations of the ideal function. The third order polynomial ($M=3$) shows a much better fit although only few data points hit the polynomial. However, when the polynomial order is raised to $M=9$, the polynomial achieves a perfect fit where each point lies on the polynomial curve. The problem is that the fitted curve oscillates wildly and gives a very poor representation of the function $\sin(2\pi x)$. This behaviour is called *over-fitting*. We can see that there is a trade-off in finding an adequate complexity of the feature model.

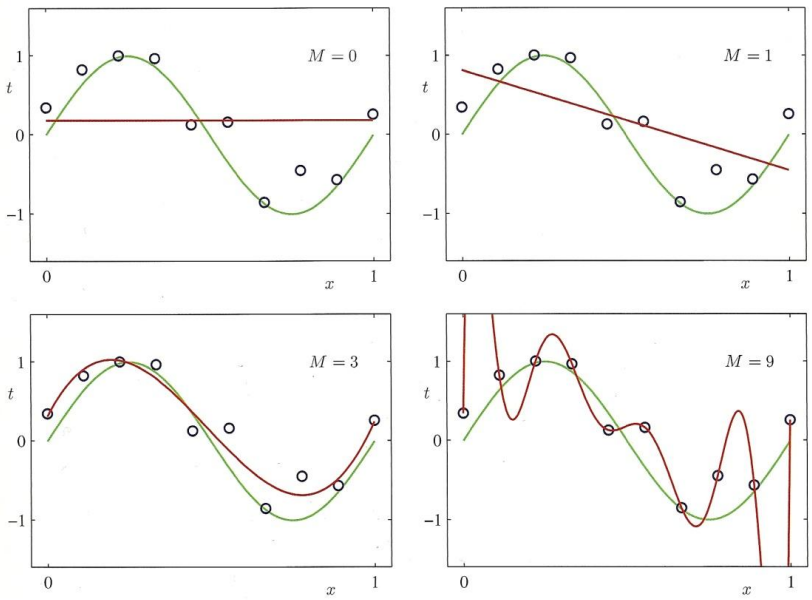


FIGURE 3.3: Illustration of the influence of model complexity on a polynomial curve fitting problem. Picture reproduced from (Bishop 2007)

Examples:

In Mandel and Ellis (2005), the authors performed experiments on artist classification on a dataset of 1200 pop songs. They compared song level features modelled with 20 Gaussian components with a single Gaussian mixture model. The classification methods used are k-nearest neighbours (kNN) and support vector machines (SVM). There are two cases for the training and testing sets: songs from separate albums and songs from the same albums. See Table 3.2 for the comparison of classification accuracies for different song-level models. Results show that the single Gaussian model outperforms the 20 Gaussian components model for both types of classifiers and collections. The poorer performance of the 20 Gaussian component model is a clear example of overfitting. Meanwhile, music genre classification experiments by Pampalk (2006a) showed that there is no significant difference in performance in using single Gaussian model compared to using 30 Gaussian components. Given that the computation using single Gaussian models is much faster than using multiple Gaussian components, then it becomes a practical model for song-level features.

TABLE 3.2: Classification accuracy for different song-level models (Mandel and Ellis 2005)

Classifier	*Song Model*	*Separate Albums*	*Same Albums*
KNN	Single Gaussian	0.524	0.722
KNN	20 Gaussians	0.365	0.515
SVM	Single Gaussian	0.648	0.839
SVM	20 Gaussians	0.431	0.365

3.2.3 Dynamics of the data

The third objective is to model the dynamics of the data. The bag-of-frames approach commonly used to model features work on a time scale defined by the frame size. The distributions of the features are derived from the collection of frames and the sequencing or time ordering of the frames is disregarded. Some of the authors try to improve their algorithms by taking the dynamics of the data into account.

- *frame size*: In short-time audio analysis, the signal is segmented into small, possibly overlapping, segments in time and each segment is processed separately. The number of samples per segment is called the frame size. The frame size must be small enough so that frequency characteristics of the magnitude spectrum are relatively stationary. The work by Alluri and Toiviainen (2009) suggests that the correlation between the features and perceptual dimensions (e.g. activity, brightness, fullness) depend on the length of the frame size used for analysis.

- *block-level processing*: In block-level processing framework, each block is composed of several frames that allows the extracted features to better capture temporal information, see Figure 3.4. Block-level feature have already been proven to be useful in automatic music genre classification, tag classification and music similarity estimation (Seyerlehner et al. 2010). For example, the feature Spectral Pattern that characterizes the frequency content is composed of the magnitude spectrum of 10 consecutive frames. In general, the number of frames used per block depends on the type of feature that is extracted.

- *time derivative*: It has been shown that the performance of a speech recognition system can be greatly improved by adding time derivatives to the basic static features (Furui 1986), see Equations 2.1 and 2.2. For music processing, improvements in genre classification accuracy were reported by de Leon and Martinez (2012a) in using the derivative coefficients of the MFCCs together with the static MFCCs as compared to using the static MFCCs alone.

- *texture window*: The texture window corresponds to the minimum length of sound that is necessary to identify a particular sound or music texture (Tzanetakis and Cook 2002). This can be derived as the running means and variances of the extracted features over a number of consecutive frames. Unlike the block-level processing framework, the texture window size is fixed for all features derived. For example, the frame size can be defined as 23 ms (512 samples at 22050 Hz sampling rate) whereas the texture window is 1 second (corresponding to 43 frames).

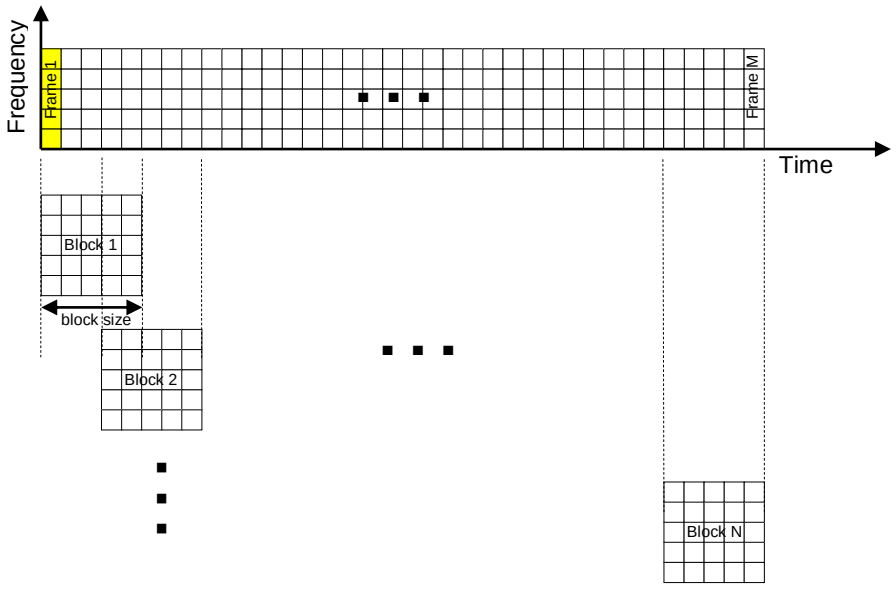


FIGURE 3.4: Block processing of the magnitude spectrum with M feature frames (Seyerlehner et al. 2010).

3.3 Experimenting with timbre features

We explore the feature space of polyphonic timbre by evaluating features individually and as a combination of features. All evaluations are done on the same computer using Matlab® R2009b, see Appendix 2 for the codes. This machine is powered by an Intel® Xeon® CPU (quad-core @ 2.67 GHz) with 12 GB RAM running 64-bit Windows 7 Enterprise.

3.3.1 Datasets

Pampalk (2006a) described that over-fitting may occur when the parameters of a model are optimized in such a way that they produce good results using specific set of data. Thus, it is common for researchers to use several datasets in their study. In addition, no single dataset can adequately represent the complexity and variability of the content-based audio retrieval task. To perform comparative evaluations of algorithms, we need to use several datasets in a consistent evaluation environment. We can then start tuning the parameters and observe if improvements occur with most datasets. If the improvement in results is consistent across datasets, then we can conclude that the new settings outperform the original.

There are four datasets used for the evaluation of timbre similarity: 1) DB-S, 2) GTZAN, 3) 1517-Artists, and 4) Unique.

1. **Small database (DB-S)**
   This dataset is composed of 250 songs: 50 tracks from five genres; each genre represented by 5 or more artists, with each artist having at most 10 tracks. The songs come from the author's personal collection and serves as a starting point for timbre similarity measures. This dataset is referred as "DB-S", see Appendix 3 for the list of music tracks in this dataset. The songs were selected by the author from five timbrally consistent, but not necessarily non-overlapping genres. The five general groups are pop rock, classical, electronica, hard rock, and hip-hop. The paragraphs below describe each music genre.

   The *pop rock* group contains songs that primarily use acoustic or electric guitars, few percussions and vocals. The tempo of the songs may vary but the mood is generally pleasant. The related genres included in this group are soft rock (The Beatles, Eraserherads), acoustic pop (Sabrina), and country pop (John Mayer, Taylor Swift).

   The *classical* group contains pieces that were composed before the 20^{th} century. The instruments used are found in a symphony orchestra, together with a few solo instruments such as the piano, harpsichord, and organ. Unlike other music groups, most classical works exhibit complexity through the use of counterpoint, thematic development, texture, etc.

   The *electronica* group contains contemporary, percussive music that is primarily designed for dancing. The songs were constructed using electronic instruments, synthesizers, sequencers, etc. Some songs may even be influenced by alternative rock or borrow segments from other music genres.

   The *hard rock* group is typified by a heavy use of distorted electric guitars, bass guitar, drums, and often accompanied with piano and keyboards. The vocals are often growling, raspy, or involve screaming. The songs in this group are generally described as loud and aggressive. The artists in this group include AC/DC, Guns n' Roses, Kiss, Metallica, and Rage Against The Machine.

   The *hip-hop* group consists of stylized heavy beat music that accompanies rapping, singing or chanted speech. It may also incorporate scratching, beatboxing, and instrumental tracks.

2. **GTZAN**
   The GTZAN dataset consists of 1000 audio files each 30 seconds long. It contains 100 tracks from 10 genres. The collection is downloaded from MARSYAS project

website[27]. There is no artist or album information given for this dataset. However, it is expected that artist effects will be observed as listening to some of the songs revealed that some artists and songs are the same. The "GTZAN" dataset is a popular dataset used in MIR literature and should help compare our results with previous studies.

3. **1517-Artists**
   This dataset consists of freely available songs[28]. It was used in Seyerlehner et al. (2008) for developing music similarity algorithms. There are 3180 tracks by 1517 different artists distributed over 19 genres. In our experiments, only 10 popular Western genres from this collection were selected, as listed in
   Table 3.3. The distribution of tracks is almost uniform across different genres.

4. **Unique**
   This dataset contains 30-second song excerpts from 3115 popular songs distributed over 14 genres (Seyerlehner et al. 2010). However, only 10 popular genres were also selected. The dataset is compiled in such a way that no two songs by the same artist are in the dataset. This dataset has an uneven distribution of the songs. For example, the smallest genre accounts for only 1.4% while the largest genre accounts for 26.9% of all songs.

TABLE 3.3: Composition of the datasets used in local experiments.

Dataset	*Genres*
DB-S	pop rock (50), classical (50), electronica (50), hip-hop (50), hard rock (50)
GTZAN	classical (100), hip-hop (100), hard rock (100), pop (100), blues (100), country (100), disco (100), jazz (100), metal (100), reggae (100)
1517-Artists	classical (125), electronica (164), hip-hop (155), hard rock (181), blues (186), country (187), jazz (177), reggae (171), alternative punk (182), rhythm and blues (174)
Unique	classical (744), electronica (187), hip-hop (229), hard rock (398), blues (42), country (58), disco (766), jazz (310), reggae (74), rhythm and blues (39)

Limitations on using the datasets

The basic methodology in this work is to apply the same methods consistently to the four datasets described above. Although the types of music in a dataset are diverse, it is obvious that they only represent a small fraction of "world music". If improvements are consistent across the range of datasets, then it can be argued that they are independent of the datasets. However, the validity of the results is limited to the music genres used.

[27] http://marsyas.info/download/data_sets
[28] www.seyerlehner.info

DB-S and GTZAN datasets are used for preliminary experiments for fast optimization of some parameters (e.g. length of the input sample, number of MFCCs, frame size). The DB-S dataset is particularly important since *artist* and *album effects* should be observed in this case. Artist or album effect means that performance results may be artificially increased since the database contains songs from the same artist or album as the query song. In the context of timbre similarity, these effects are actually desired as they validate the measure of timbre similarity. For example, if the query song is a Beatles song and the music collection contains other Beatles songs from the same studio album, then it should be expected that they are the most similar songs.

3.3.2 Audio format

The formats of the audio files to be processed are adopted from the formats used at the Audio Music Similarity and Retrieval Task of MIREX[29]. The sampling rate is 22050 Hz, with a sampling size of 16-bit, single channel (mono), and encoded in WAV format. A 30-second clip is from the middle of each song is used.

The parameters are chosen to reduce the number of data to be processed without losing too much of the important information. For example, the CD-quality audio standard has 2 channels (stereo) sampled at 44100 Hz. A 1 second CD-quality audio clip is then represented by 88200 data points. Using our parameters, the same clip can be represented by just 22050 data points. A stereo file can be converted to mono by getting the average of the two channels whereas the sampling rate can be changed by downsampling or subsampling the data. Lastly, the audio clip is selected from the middle of each track to avoid lead-in and fade-out effects. It is assumed that the most important parts of a song (e.g. chorus) will likely occur in this region.

[29] http://www.music-ir.org/mirex/wiki/2011:Audio_Music_Similarity_and_Retrieval

The following Matlab script converts an MP3 file into the desired WAV format:

```
%decode MP3 file into PCM signal, extract 30 s clip from middle
%save as audio object
audio_object = miraudio('sample.mp3','Extract',-15,15,'s','Middle');
%get raw audio data
audio_raw    = mirgetdata(audio_object);
%reduce sampling rate from 44100 Hz to 22050 Hz
audio_resample = decimate(b,2);
%save resampled audio signal as WAV file
wavwrite(audio_resample,22050,'sample.wav')
```

where `miraudio( )` is a function from MIRToolbox (Lartillot and Toiviainen 2007). As a consequence of subsampling to resample the signal, aliasing is produced as discussed in the latter part of Section 3.1.

3.3.3 Evaluation measures

This section discusses how to measure the performance of an audio retrieval system. The text retrieval community has been using evaluation measures over the years and has well established methodologies (Salton 1992). This section enumerates some well-known methods, including their limitations and justification of their use in this work.

First, it is worthwhile to understand the task of audio retrieval so as to lay down the context of what the evaluation measure must attempt to quantify. For example, a user inputs a query song in an audio retrieval system. Then the system returns a list of songs that is estimated as relevant to the query. The user may then listen to music clips or possibly select some clips and search again.

In this example, we can see that the number of songs that are returned is an important factor in evaluation measures. While it is possible to return a ranked list of all the songs in a database, given that there may be millions of tracks in that dataset, a user will most likely browse only a small number of songs. The first few songs on the top of the list are particularly significant. Thus, we could limit measuring the retrieval performance up to the top 20 of the returned songs.

There are other issues for real systems and searches. The number of relevant music in a dataset will affect any measure. In an experimental set-up this will be known, while in a real environment, this may be unknown. It is possible that a user may be searching for a

particular song; in which case, only one song is relevant. In other scenarios, there may be thousands of relevant songs, or in the worst case, even none.

We need to find measures that give a good indication of music search effectiveness. These would enable us to compare algorithms that are being studied and check how the results fit user requirements. In the report by Cleverdon and Keen (1966), he came up with six areas to quantify the performance of a document search system. These are coverage of the dataset, presentation of results, user effort, time lag between query and answer, precision, and recall. Among these measures, we use the most relevant areas applicable to this study, which are the time lag between query and answer, precision, and recall.

3.3.3.1 Precision and recall

Precision and *recall* are the basic measures used in information retrieval evaluation. Precision is a measure of the ability of a system to present only relevant items.

$$precision = \frac{number\ of\ relevant\ items\ retrieved}{total\ number\ of\ items\ retrieved} \tag{3.6}$$

Recall is a measure of the ability of a system to present all relevant items.

$$recall = \frac{number\ of\ relevant\ items\ retrieved}{total\ number\ of\ relevant\ items} \tag{3.7}$$

Precision-recall graphs

If we calculate precision and recall for a list of returned items, the values will be independent from the ordering of the list. Normally we are interested in retrieving relevant items first. To get this information, we need to calculate both precision and recall as the length of the result list is varied. This result is often shown using a precision-recall graph. The graph can easily characterize a search engine, as well as the underlying algorithms used. Consider the graph shown in Figure 3.5. The characteristics of the curves are quite different. System A has higher precision at low recall. This is more

appropriate in multimedia search engines where the user wants as many relevant items in the first few results. System B has higher precision at high recall. This is more suited for a user who wants to ensure that all the relevant items are retrieved. As such, these two systems clearly define two different algorithms.

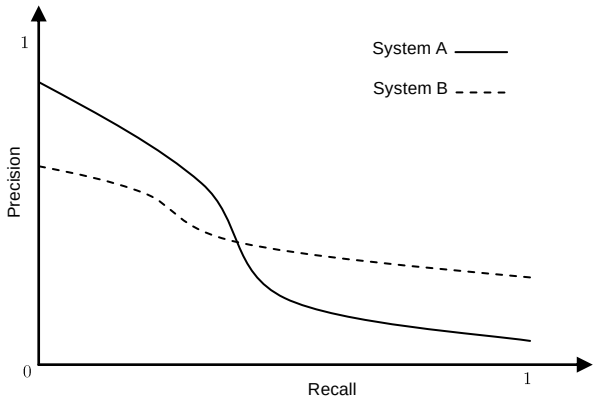


FIGURE 3.5: Example Precision-Recall graphs of two retrieval systems.

To construct a precision-recall graph, precision is plotted against recall as we step through the retrieved document list. Suppose a database has 10 items and the query has 4 relevant items that are 1st, 3rd, 5th, and 6th in the ranked results list. This scenario is shown in Table 3.4 where the relevant items are in bold. Using equations 3.6 and 3.7, we start with just the first item in the result list; this is relevant so precision is 1 and recall 0.25. The next item is non-relevant, precision drops to 0.5 and recall stays at 0.25. The third item is relevant and precision is now 0.67 with recall 0.5. The next item is non-relevant; recall stays the same, but precision drops to 0.5. The next two items are relevant; precision increases to 0.6 then 0.67, and recall increases to 0.75 then 1. Going further down the list, precision continues to decrease while recall settles at 1.

TABLE 3.4: Example calculation of precision and recall

	Depth of retrieval list									
	1	2	**3**	4	**5**	**6**	7	8	9	10
Precision	1	0.5	0.67	0.5	0.6	0.67	0.57	0.5	0.44	0.4
Recall	0.25	0.25	0.5	0.5	0.75	1	1	1	1	1

To facilitate computing average performance over a set of queries, each with a different number of relevant documents, individual query precision values are interpolated to a set of standard recall levels (0 to 1 in increments of 0.1). This is called *interpolated precision*. The particular rule used to interpolate precision at standard recall level i is to use the maximum precision obtained for the query for any actual recall level greater than or equal to i.

Average precision

Precision and recall are single value metrics based on the whole list of items returned by the system. For systems that return a ranked list of items, it is desirable to also consider the order in which the returned items are presented. This can be measured with the average precision:

$$average\ precision = \frac{\sum_{r=1}^{K} precision\ (r)}{K} \tag{3.8}$$

where *precision(r)* is the precision when each of the K relevant items is retrieved. Using the example in Table 3.4, the average precision is $\frac{1+0.67+0.6+0.67}{4} \approx 0.74$. High values of average precision mean that most of the relevant items are on top of the list. For example, suppose there is a single relevant item that is returned in the top ten results. If the relevant item appeared first, then the average precision is 1. If the relevant item appeared last, then the average precision is 0.1.

Given a set of queries, we can take the mean and this metric becomes the mean average precision. The mean average precision gives the overall performance, but its value may hide interesting patterns due to the averaging process.

Precision at *n* items

This is the precision measured when a specific number of items are retrieved. The choice of what level to use depends on the task. As described in Section 3.3.3, we may choose n

as the reasonable number of songs that a user may wish to preview. In our evaluations, we measure the precision at 5, 10, 15, and 20 items.

A limitation of using this measure is that it is sensitive to the number of relevant items in the dataset. For example, a query with only 1 relevant item could only achieve a maximum precision at 100 of 0.01; while if there are 100 relevant items, the maximum precision at 100 is 1.

R-precision

R-precision is a useful measure to get around the disadvantage of using precision at n. It is defined as the precision at the R-th item retrieved, where R is the number of relevant items for that query. It can be used for comparing the performance of different features or algorithms for a particular query. For example, a query with only 1 relevant item should be evaluated using precision at 1; if there are 100 relevant items, the precision at 100 should be used.

***F*-measure**

The F-measure is the harmonic mean of precision and recall.

$$F = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{3.9}$$

This has a high value when both precision and recall are high. There is really no intuitive meaning of F measure other than it is just a combined metric. This measure can also be used to compare the performance of the algorithms.

Discussion of precision and recall

Precision and recall measure different aspects of the performance of a search engine. Although they are widely used in information retrieval, they are also subject to several limitations. One concern is that to calculate recall correctly, it is necessary to know all the relevant items in the dataset. This is not an issue if all the items in the dataset are completely and accurately tagged, which is almost impractical in reality. This means that all music tracks should be tagged with all the relevant terms by music experts. This

process takes so much time and effort. A solution is to crowdsource the tags, i.e. unmonitored users create the tags for each track. However, this may even lead to a more confused labelling of tags.

Another issue is that given the evaluations are performed using a fixed set of queries and pre-defined genre associations, the evaluation measures do not capture the degree of user satisfaction. For example, they do not measure how many times a user has to query and browse through the dataset until his requirements are satisfied. It is still possible to use the measures for manual tests where users query the system and identify relevant items. However, the subjective nature of relevance will also have to be considered. What is relevant to one person may not be relevant to another.

Moreover, the degree of relevance judgments is not strictly binary. Given a set of relevant items, some may be more relevant than others for a user. This infers that high precision or recall values may not be necessary if the user is searching for only one particular item.

In this work, we are focused on algorithms for feature extraction, similarity estimation, and indexing. It is beyond the scope of the study to consider user preferences. For this reason, we can still use precision and recall for evaluations. Generally, we use the average precision at n items. Over a wide range of queries, this generally gives a good measure of performance for an audio retrieval task. Precision and recall are also widely understood and adopted by other researchers. More importantly, these measures are also used in the annual MIREX[30].

3.3.3.2 Presence of hubs and orphans

Most techniques for timbre similarity are based on spectral similarity measures. However, these are known to exhibit undesirable properties such as the existence of *hubs* and *orphans*. Hubs in music datasets were first discussed by Aucouturier and Pachet (2004). A hub is a song that is 'always similar', based on the computational model of similarity,

[30] http://www.music-ir.org/mirex/wiki/2013:Audio_Music_Similarity_and_Retrieval

to a large number of other songs. This computational similarity does not correspond to perceptual similarity. Closely related to the analysis of hubs is the presence of orphans. An orphan is 'never similar' to any other piece despite having perceptual similarity to a large number of songs. This case was first discussed in Pampalk's work (Pampalk 2006b).

Karydis et al. (2010) proposed a conceptual framework to provide an explanation of the conditions that create *hubness*. The framework is based on the concentration property in high-dimensionality spaces. This property refers to the tendency of all points in a high-dimensional feature space to be almost equally distant from each other. The study focused on timbre similarity and the main parameter examined is the number of MFCCs that defines the dimensionality of the feature space.

From their experiments, it was observed that the distribution of distances at any reference point has a finite variance for a given dimensionality value. In the case of using more coefficients, the items closer to the centre have a tendency to become closer to all other points in the feature space. This increases the probability of hubs as the items are also near neighbours to many other points. On the other hand, it is also expected to have a number of points farther from the centre, forming orphans. Therefore, there is a trade-off between using a higher number of coefficients to better characterize timbre and the probability of forming hubs and orphans.

A consequence of *hubness* is that many nearest neighbour relations in the distance space are asymmetric, i.e. object x is amongst the nearest neighbours of y but not vice versa. Two methods have been evaluated by Schnitzer et al. (2012) that try to symmetrise nearest neighbours relations. The first method is a local scaling method that uses the average distance of the k nearest neighbours. This is called the non-iterative contextual dissimilarity measure (NICDM) (Jegou et al. 2010) and is defined by

$$NICDM(d_{x,y}) = \frac{d_{x,y}}{\sqrt{\mu_x \mu_y}} \tag{3.10}$$

where μ_x and μ_y denote the average distance to the k nearest neighbours of object x and y respectively.

The second method that the authors proposed is called Mutual Proximity (MP). It is a global scaling method that is based on: 1) transforming a distance between x and y into a probability that y is the closest neighbour to x given the distribution of the distances of all points to x in the database, and 2) combining these probabilistic distances from x to y and vice versa via their joint probability.

$$MP(d_{x,y}) = P(X > d_{x,y} \cap Y > d_{y,x}) \tag{3.11}$$

Empirical results show that both local and global methods perform at about the same level. They can reduce hubness effectively, especially for datasets of high dimensionality that are most affected by hubness. The methods however have little impact on datasets with low dimensionality.

In this work, it will be shown that normalizing the distance values between songs into standard scores is an effective way of reducing the hubs, see Section 3.5.

3.3.3.3 Genre classification accuracy

The task of audio similarity estimation is closely related to automatic genre classification. Given an untagged song and a ranked list of similar songs, we can use k-nearest neighbours (kNN) approach for genre classification. The main advantage of using kNN is its simplicity. Its classification is based on majority vote among an object's k nearest neighbours. The object is classified to the genre most common amongst its neighbours. However, its disadvantage is that no distinction is made if all objects close to the query are perceptually similar.

For example, let the parameter k be set to 5 and that three of the five nearest neighbours to a query are *pop*. Based on majority rules, the query song is labelled as *pop*. It does not necessarily mean that all the nearest neighbours sound alike since it is possible that only one of them is perceptually similar to the query.

3.3.3.4 Response Time

The response time of a search engine is very important to the users. The response time or time lag is equivalent to the average interval between the time the search request is made and the time an answer is given. Although this is dependent on many variables, it will give a realistic performance measure after repeated experiments in the same conditions, and the lag times are averaged.

If we are considering the speed of the retrieval engine we can look at the individual elements that make up the total response time. First, let us consider the effect of response times on user perception. The general advice regarding response times for interactive systems have been described in Miller (1968).

- **0.1 second** is about the limit for having the user feel that the system is reacting instantaneously. No special feedback is necessary except to display the result.
- **1.0 second** is about the limit for the user's flow of thought to stay uninterrupted, even though the user will notice the delay. The user does not lose the feeling of operating directly on the data.
- **10 seconds** is the limit for keeping the user's attention focused on the task. The users should be given feedback indicating when the computer expects to be done. If the response time is longer than this it is likely that the user will start doing something else during the gaps.

Given these timescales the question now is what is an acceptable search time? A user would expect an audio search to take some time as the system has to process the data. Therefore the response time should be between 1 to 10 seconds. With a web-based application there would also be a small overhead on the presentation of results. We therefore use this time allowance as our target for the time lag between query and response.

Factors affecting the response time

One way to break down the total response time is to profile each individual component of the block diagram in Figure 2.2. We can then consider the time taken for each component: query submission, feature generation, loading features, similarity comparison,

ranking of results and presentation of results. Profiling can also be done for every program functions used by each component. This is helpful to identify the functions that consumes the most time, and therefore find ways to improve performance.

When searching a large dataset, similarity computation (i.e. loading the required features and comparing them) consumes the most time. As a dataset grows, there will be a size where it becomes impractical for the features to be stored in a computer's memory and it will then need to be stored on and loaded from high capacity drives such as hard disk drive (HDD) or solid state drive (SSD). If HDD is used, the disk access time will become the bottleneck in the system as it is orders of magnitude greater than memory access time (Hennessy and Patterson 2006).

Another factor that significantly affects the response time is the complexity of the matching algorithm. Suppose a dataset contains N data items and it is desired to retrieve the k nearest items to a query song. A naive implementation would compare each item to the query song then rank the results. This means that the number of operations is $O(N)$ or linear time. Since commercial databases usually contain millions of music tracks, it may take a very long time to complete the operation, especially if the matching algorithm is complex. This is an area that needs to be improved.

For the purposes of this thesis, the primary speed measure used is wall-clock time, i.e. actual time taken by a computer to complete a task[31]. This simple approach provides a useful measure. Indeed, the system should satisfy the user needs and work within the bounds of the recommended response time. In addition, we will provide user feedback on the progress of each process in the system.

3.4 Experiments and results

The development of the timbre similarity algorithm is done in stages. Initial evaluation and modifications to the features are performed using the DB-S and GTZAN music

[31] http://www.catb.org/jargon/html/W/wall-time.html

datasets. After the general parameters are obtained, evaluations of each feature set are carried out using all four music datasets discussed in Section 3.3.1. This is followed by combining the features to improve the algorithm's overall performance.

The main evaluation measure used throughout the work is precision at n items. We also quantify the presence of *hubs* and *orphans* in the results. Lastly, we plot a sample histogram to show how many times a song in a dataset appears in a top 5 list.

3.4.1 Optimization of features

The features presented in Chapter 2 come with a large set of parameters. However, it is impractical to perform a complete evaluation using all possible parameter variations. Each parameter setting requires evaluating the feature on the whole dataset. We try to optimize each feature by finding the general parameters of the MFCC feature space. Then we apply the general parameters to all other features. This reduces the number of variables needed in optimizing other features.

MFCC

The first experiment is to determine the general parameters to be used in succeeding experiments. This focuses on the MFCC feature space as this is one of the most commonly used features in MIR. This preliminary experiment uses the DB-S and GTZAN datasets. We explore the timbre space by varying the following parameters:

- frame size on which we compute the MFCCs {512, 1024, 2048 samples}
- number of MFCCs {12, 14, 16, 18, 20 coefficients}
- inclusion/exclusion of MFCC's 0^{th} coefficient: correlated with the signal's energy

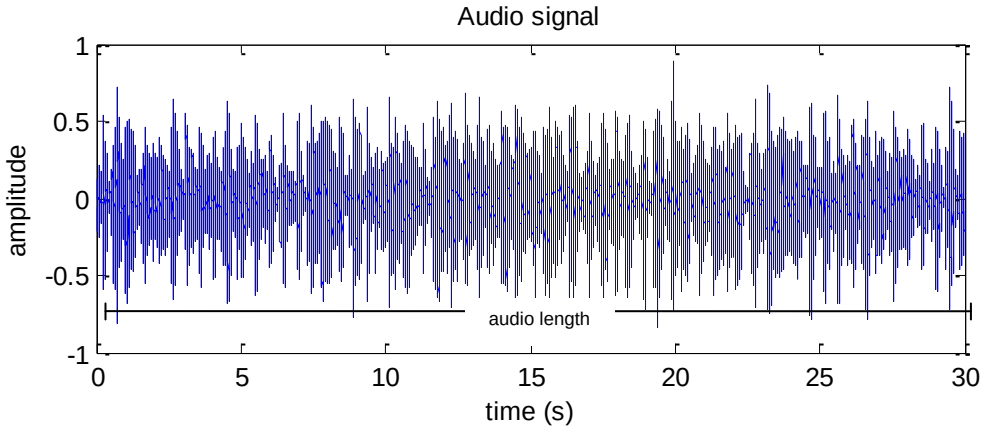
The parameters above can have a significant impact on the information captured and on subsequent search performance. As mentioned before, timbre is a feature that can be described using a short audio clip (e.g. less than a second) unlike other features like rhythm that requires longer audio sample. Therefore we need to choose an audio length or frame size over which to calculate the feature, see Figure 3.6. If the frame size is too short, then important information may be lost. If the frame size is too long, then the

signal may not be pseudo-stationary, i.e. statistics are homogenous. Note that the frame size values are powers of 2 (e.g. $512=2^9$) to facilitate fast computation of Fourier transform using the Cooley-Tukey algorithm (Cooley and Tukey 1965). The minimum frame size is 512 since using 256 samples causes numerical errors when the covariance matrix of the features is computed.

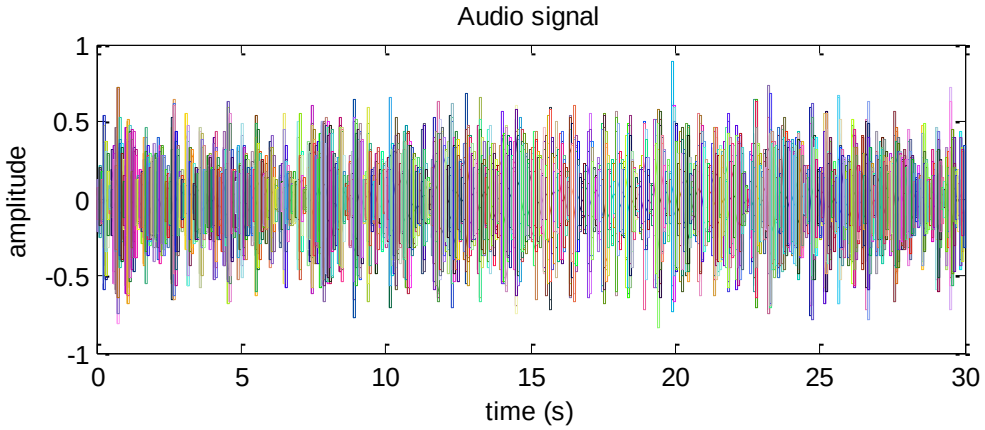
The number of MFCCs determines the precision by which the signal's spectrum is approximated, which implies more variability in the data. The trade-off is that more MFCCs lead to greater risk on the effects of the curse of dimensionality. Thus, we set the upper bound on the number of MFCCs to 20 as recommended in (Mandel and Ellis 2005). Another issue is whether to include the 0^{th} coefficient or not. The 0^{th} coefficient is the average of the logarithm of the summed contents of the Mel-frequency triangular bands. Thus it can be interpreted as the loudness averaged over the triangular bands. On the one hand, loudness may be useful for modelling a song. On the other, the original loudness of the song had already been lost after normalizing the signal in the time domain as a pre-processing step.

We computed precisions at 5, 10, 15, and 20 items $\{P_5, P_{10}, P_{15}, P_{20}\}$ for all the songs in a database and computed their average. Table 3.5 shows the effects of varying the general parameters on the DB-S dataset. Table 3.6 shows the variance of the precisions. The following are the observations based on the results:

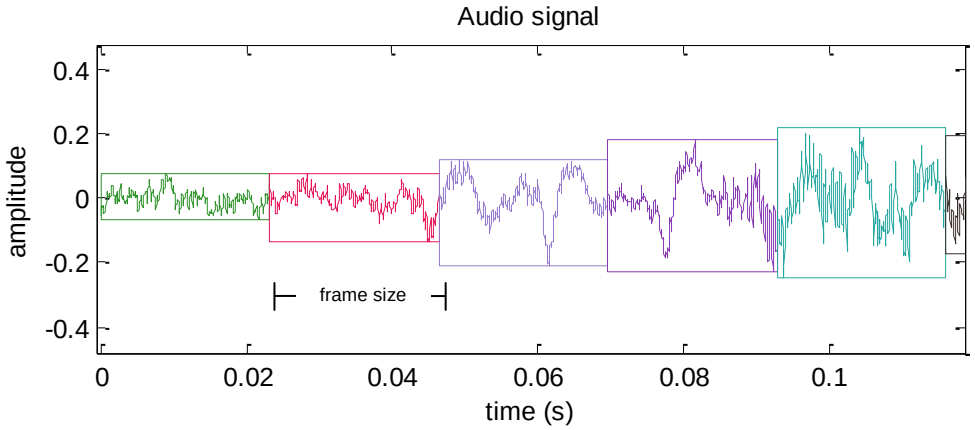
1. By varying the general parameters, the difference between the best and worst precisions for the DB-S dataset is at least 10%; for the GTZAN dataset the difference is at least 15%.
2. The best parameters are the following: frame size of 512 samples; number of dimensions of MFCC is 20 inclusive of the 0^{th} order coefficient.
3. Increasing the number of MFCC dimensions improves the precisions whereas increasing the frame size decreases the precisions.
4. The variance of precisions for a frame size of 512 samples is comparable with the minimum value.



(a) 30-second audio clip



(b) Audio clip segmented into frames



(c) Frame-level audio clips

FIGURE 3.6: Frame-level audio segmentation for feature extraction.

Results in Table 3.5 show that it is possible to improve the precisions by varying general parameters of the MFCC feature space. The best configuration for the MFCC feature space is then considered as the *baseline* to which the other feature spaces are compared.

We then set the frame size to 512 samples (23 ms at 22050 Hz sampling rate) as the default setting for all other features.

TABLE 3.5: Retrieval performance using MFCCs for varying number of coefficients and frame size.

Dimensions	*Precision at 5*			*Precision at 10*		
	Frame size			Frame size		
	2048	1024	512	2048	1024	512
20	0.735	0.74	**0.748**	0.735	0.74	**0.748**
18	0.737	0.74	0.744	0.737	0.74	0.744
16	0.736	0.736	0.74	0.736	0.736	0.74
14	0.734	0.738	0.74	0.734	0.738	0.74
12	0.722	0.727	0.73	0.722	0.727	0.73
	Precision at 15			Precision at 20		
	Frame size			Frame size		
	2048	1024	512	2048	1024	512
20	0.708	0.71	**0.716**	0.69	**0.691**	0.69
18	0.708	0.712	0.711	0.689	0.687	0.689
16	0.705	0.709	0.712	0.684	0.683	0.684
14	0.705	0.706	0.71	0.684	0.685	0.682
12	0.685	0.689	0.692	0.663	0.665	0.665

TABLE 3.6: Variance of precisions for varying number of MFCCs and frame size.

Dimensions	Precision at 5			Precision at 10		
	Frame size			Frame size		
	2048	1024	512	2048	1024	512
20	0.109	0.101	0.078	0.095	0.088	0.078
18	0.109	0.098	0.084	0.095	0.088	0.084
16	0.108	0.098	0.082	0.089	0.086	0.082
14	0.103	0.098	0.081	0.088	0.086	0.081
12	0.099	0.098	**0.077**	0.085	0.081	**0.077**
	Precision at 15			Precision at 20		
	Frame size			Frame size		
	2048	1024	512	2048	1024	512
20	0.088	0.081	**0.079**	0.085	**0.077**	0.078
18	0.088	0.086	0.085	0.085	0.084	0.084
16	0.086	0.083	0.083	0.083	0.083	0.082
14	0.086	0.082	0.081	0.082	0.083	0.081
12	0.081	**0.079**	**0.079**	**0.077**	0.078	**0.077**

Delta MFCCs and Delta-Delta MFCCs

The first and second derivatives of the Mel-frequency cepstral coefficients can be appended to each MFCC feature vector. This results in a three-fold increase in feature dimension. The time derivatives account for the dynamics of the features. The assumption is that appending these features will result in an overall improvement in the precisions, similar to the improvements of performance in speech recognition systems. For the next experiments, we evaluate the time derivative features individually. We vary the length of the time window used in computing the derivatives, see Equations 2.1 and 2.2. We then append the time derivatives to the MFCCs using the best time window. The results are tabulated in Table 3.7 and Table 3.8.

TABLE 3.7: Summary of precision using Delta MFCCs

Window size (frames)	*DB-S*				*GTZAN*			
	P_5	P_{10}	P_{15}	P_{20}	P_5	P_{10}	P_{15}	P_{20}
3	**0.71**	**0.67**	**0.64**	**0.62**	**0.61**	**0.55**	**0.51**	**0.49**
5	0.7	0.66	0.63	0.61	**0.61**	0.54	0.5	0.48
7	0.68	0.65	0.61	0.59	0.6	0.54	0.5	0.47
9	0.66	0.63	0.6	0.57	0.59	0.53	0.49	0.46
11	0.65	0.62	0.59	0.57	0.58	0.53	0.49	0.46

TABLE 3.8: Summary of precision using Delta-delta MFCCs

Window size (frames)	*DB-S*				*GTZAN*			
	P_5	P_{10}	P_{15}	P_{20}	P_5	P_{10}	P_{15}	P_{20}
3	0.68	**0.66**	**0.64**	**0.63**	**0.6**	**0.54**	**0.51**	**0.48**
5	**0.69**	0.65	**0.64**	0.62	**0.6**	**0.54**	0.5	0.47
7	0.68	0.65	0.63	0.61	0.59	0.53	0.5	0.47
9	0.68	0.64	0.61	0.6	**0.6**	0.53	0.49	0.46
11	0.68	0.64	0.61	0.59	0.59	0.52	0.49	0.46

The results show that the best time window to use is 3 frames. Hence, we use this configuration as the time derivatives are appended to the MFCC feature vectors. The result of this experiment is shown in Table 3.9.

TABLE 3.9: Summary of precision using MFCC appended with its time derivatives

Features	*DB-S*				*GTZAN*			
	P_5	P_{10}	P_{15}	P_{20}	P_5	P_{10}	P_{15}	P_{20}
MFCCs	0.79	0.75	**0.72**	0.69	0.63	0.56	0.52	0.48
MFCCs:Delta	**0.8**	**0.76**	**0.72**	**0.7**	**0.68**	**0.61**	**0.56**	**0.52**
MFCCs:Delta:Delta-delta	0.79	0.75	**0.72**	**0.7**	**0.68**	0.60	**0.56**	**0.52**

Appending the MFCC feature vectors with its time derivates produced mixed results. There is no significant improvement using the DB-S dataset whereas there is an average increase of 8% in the precision using the GTZAN dataset. This is unexpected since the derivatives have been successfully applied to speech processing. We can also see that there is no difference in the performance in using just the first derivative compared to using both the first and second derivatives.

The observations above highlight the inadequacy of the time derivatives in capturing the dynamics of polyphonic timbre. This is also supported in (Alluri and Toiviainen 2009) where it was found that the delta MFCCs have low correlations with the perceptual dimensions of polyphonic timbre. Unlike speech, polyphonic timbre contains asynchronous instruments or voices, i.e. one instrument may be present in one frame but absent in the next. The time derivatives of the MFCCs measure the trajectories of the coefficients over time at fixed time intervals. If the sound sources are changing asynchronously over time, as in the case of polyphonic timbre, then the time derivatives fail to capture these changes. Hence, the delta and delta-delta coefficients are no longer used for the remainder of this work.

Spectral contrast

The main parameters for spectral contrast are the neighbourhood factor and the size of the filter bank. We set the neighbourhood factor at 0.2, see Equations 2.3 and 2.4. We then evaluate the effect of varying the size of the filter bank. This parameter determines the dimensionality of this feature. In the original algorithm, the size of the filter bank was set to 10 filters. Since our implementation is more similar to MFCC feature

extraction, we vary the size of the filter bank from 12 to 20 filters. Table 3.10 lists the results.

TABLE 3.10: Summary of precision using spectral contrast

Size of filter bank	*DB-S*				*GTZAN*			
	P_5	P_{10}	P_{15}	P_{20}	P_5	P_{10}	P_{15}	P_{20}
20	**0.77**	**0.72**	**0.69**	**0.66**	0.54	0.49	0.45	0.42
18	0.76	**0.72**	**0.69**	**0.66**	0.54	0.48	0.44	0.41
16	0.75	0.70	0.67	0.66	**0.55**	0.48	0.45	0.42
14	0.75	0.70	0.67	0.65	**0.55**	**0.50**	**0.46**	**0.43**
12	0.72	0.68	0.66	0.64	**0.55**	0.48	0.45	0.42

For the DB-S dataset, the best precisions are obtained when the size of filter bank is 20. However, there are no significant changes in the precisions when the size of the filter bank is varied using the GTZAN dataset. Considering the benefits of using 20 filters on the DB-S dataset, we set the size of the filter bank to 20.

Sub-band flux

For sub-band flux, the main parameter is the size of the filter bank. The original algorithm by Alluri and Toiviainen (2009), which we also adopt, used 10 octave-scaled filters. We also evaluate the effect of varying the size of the filter bank. Table 3.11 lists the results.

The best results are obtained using 10 filters. For the DB-S dataset, the differences in the precisions are insignificant. For the GTZAN dataset, the effect of varying the size of filter bank is more obvious. Consider the case of decreasing the size of the filter bank from 10 to 9. This change has almost no effect on the results for the DB-S dataset. However, the precision decreased by an average of 5% using the GTZAN dataset. Thus, we set the size of the filter bank to 10.

TABLE 3.11: Summary of precision using sub-band flux

Size of filter bank	*DB-S*				*GTZAN*			
	P_5	P_{10}	P_{15}	P_{20}	P_5	P_{10}	P_{15}	P_{20}
10	0.71	**0.66**	**0.61**	**0.58**	**0.54**	**0.48**	**0.44**	**0.41**
9	**0.72**	**0.66**	**0.61**	**0.58**	0.51	0.46	0.42	0.39
8	**0.72**	**0.66**	**0.61**	**0.58**	0.48	0.44	0.4	0.38
7	0.7	0.65	0.6	0.57	0.49	0.44	0.4	0.37
6	0.7	0.64	0.6	0.56	0.47	0.42	0.39	0.37

Spectral distribution descriptors

The spectral distribution descriptors are mostly based on the moments and characteristics of the magnitude spectrum. Collectively, they may provide an alternative representation of the spectral envelope. In the next experiment, we compare two feature spaces against the performance of spectral envelope represented by MFCCs. The first feature space is composed of the spectral distribution descriptors alone. The second is composed of MFCCs appended with the spectral distribution descriptors. This approach is adopted from the methodology used for delta coefficients. Table 3.12 lists the results.

TABLE 3.12: Summary of precision using spectral descriptors

Features	*DB-S*				*GTZAN*			
	P_5	P_{10}	P_{15}	P_{20}	P_5	P_{10}	P_{15}	P_{20}
MFCCs	0.79	0.75	0.72	0.69	**0.63**	**0.56**	**0.52**	0.48
Spectral Distribution Descriptors	**0.81**	0.76	**0.73**	**0.71**	0.53	0.49	0.45	0.43
MFCCs:Spectral Distribution Descriptors	**0.81**	**0.77**	**0.73**	**0.71**	**0.63**	**0.56**	**0.52**	**0.49**

Using the spectral distribution descriptors (9-dimensional), the results improved by an average of 2% than the best MFCC configuration (20-dimensional) using the DB-S dataset; but decreased by an average of 13% using the GTZAN dataset. Thus, the idea of replacing MFCCs by the spectral distribution descriptors is inconclusive.

The combined feature space of MFCC and spectral distribution descriptors (29-dimensional) consistently performed better than MFCC alone using the DB-S dataset.

The average improvement in precision is around 2.4%. However using the GTZAN dataset, there is no improvements in the results in general. By appending the two feature spaces, the dimensionality of the new feature space is increased. This also increases the precision of the feature space that may explain the improvement in the results. Despite the possible negative impacts of the increase in dimensionality, we are more interested in the potential benefits of using this new feature space. We denote the new feature space as *spectral shape*.

Summary of single feature spaces

We have determined the optimum parameters for each feature. We compare the performance of each feature space using all the datasets. Table 3.13 lists the summary of precisions for all the datasets.

TABLE 3.13: Summary of precision using single feature spaces

Features	DB-S				GTZAN			
	P_5	P_{10}	P_{15}	P_{20}	P_5	P_{10}	P_{15}	P_{20}
MFCC (baseline)	0.79	0.75	0.72	0.69	**0.63**	**0.56**	**0.52**	0.48
Spectral contrast	0.77	0.72	0.69	0.66	0.54	0.49	0.45	0.42
Sub-band flux	0.71	0.66	0.61	0.58	0.54	0.48	0.44	0.41
Spectral shape	**0.81**	**0.77**	**0.73**	**0.71**	**0.63**	**0.56**	**0.52**	**0.49**
	1517-Artists				Unique			
MFCC (baseline)	**0.40**	0.36	0.34	0.33	**0.39**	0.37	0.36	0.35
Spectral contrast	0.36	0.33	0.31	0.30	0.37	0.35	0.34	0.33
Sub-band flux	0.32	0.30	0.28	0.28	0.33	0.32	0.31	0.31
Spectral shape	**0.40**	**0.37**	**0.35**	**0.34**	**0.39**	**0.38**	**0.37**	**0.36**

Overall, the features that represent the spectral shape performed better than the other features. They produced the highest precision at any given number of returned results. This is closely followed by MFCCs. In the study by Jiang et al. (2002), they showed that spectral contrast performs better than MFCCs in music genre classification task. Furthermore, the study by Alluri and Toiviainen (2009) claims that MFCCs do not seem to contribute significantly to any perceptual dimensions of timbre. Our results show that

for audio similarity retrieval task, MFCCs outperform spectral contrast and sub-band flux.

In this study, we assume that the features represent a specific attribute of timbre: 1) Mel-frequency Cepstral Coefficients and spectral descriptors to model the spectral shape, 2) spectral contrast to describe the range between tonal and noise-like character, and 3) sub-band flux to describe the temporal unfolding and shaping of sound spectra. We can build a spatial timbre space using the three attributes. We applied principal component analysis (PCA) on each feature space to reduce their dimensionality to a single dimension. The resulting timbre space is shown in Figure 3.7. We then map the music clips on this timbre space. We use the DB-S database in this example. Note that this is a simple visualization of the timbre space and the spatial relationships between the music tracks may not accurately describe their true perceptual similarities. However, it may give us an idea of any patterns or clustering that might be expected among the tracks.

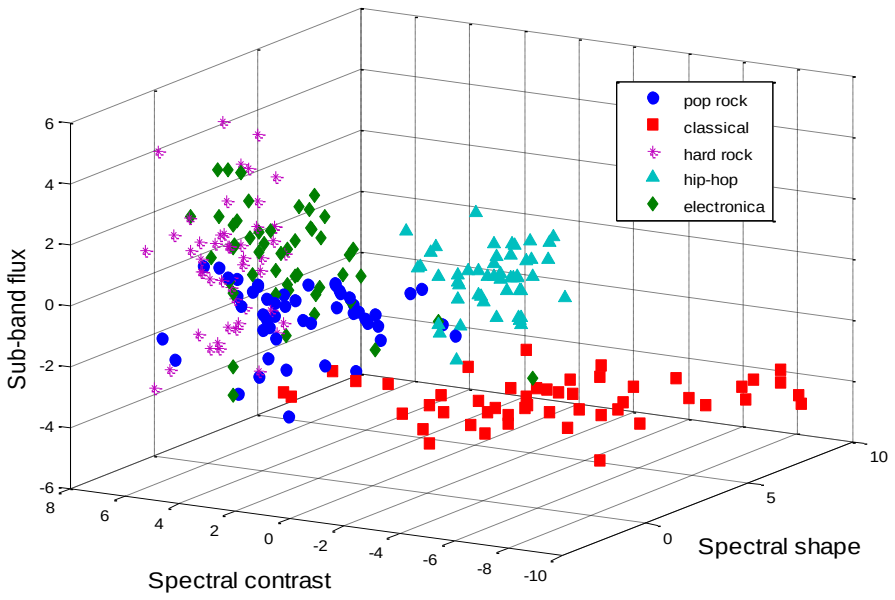


FIGURE 3.7: Mapping of the 250 DB-S music clips into the three-dimensional polyphonic timbre space.

Figure 3.7 shows some clustering of music tracks. The most obvious clustering involves the hip-hop group (cyan triangles). The tracks have small variance with respect to sub-

band flux and spectral contrast. Another obvious clustering involves the classical group (red squares). The classical tracks are widely scattered with respect to spectral contrast and spectral shape but stay below low values of sub-band flux. The hard rock group (magenta asterisks) has small variance in spectral contrast and stays at low values of spectral shape and high values of sub-band flux. The pop rock group (blue circles) is concentrated around a region near the hard rock group. Meanwhile the electronica tracks (green diamonds) occupy almost the same region as the pop rock and hard rock groups. These clusters indicate that if we use the three feature spaces, the timbre similarity algorithm should perform particularly well with the hip-hop and classical tracks of the DB-S dataset. Conversely, timbre similarity algorithm using the three feature spaces is expected to perform with less precision for pop rock, electronic, and hard rock tracks because of the clustering of the groups in Figure 3.7.

In Table 3.13 we have seen the overall performance of each single feature space on all the test datasets. We now take a closer look on the performance with respect to different music genres. Figure 3.8 displays the precision at 5 per genre, and per single feature space using the DB-S dataset. The sub-band flux feature performs best in pop rock, classical and electronic genres although the difference in performance is not significant. However, it performs significantly worse in hard rock and hip-hop genres. The spectral shape performed slightly better than using MFCCs alone except in pop rock tracks. Meanwhile the performance of the spectral contrast is comparable with MFCCs except in classical and electronica tracks.

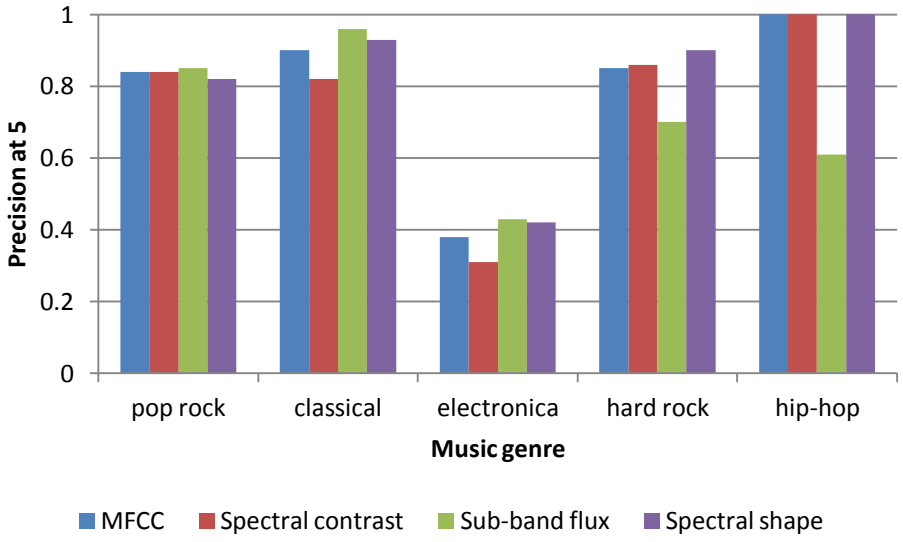


FIGURE 3.8: Precision at 5 per genre, per single feature space using the DB-S dataset.

We make the same comparisons for the feature spaces using the GTZAN dataset, see Figure 3.9. The advantage of the GTZAN dataset is that it covers a wider array of music genres than the DB-S dataset. Similar to the DB-S dataset, the spectral shape performed consistently well across all genres. The sub-band flux shows a clear advantage for jazz tracks but performed worst in several genres (disco, hip-hop, pop, reggae). The spectral contrast performed best in metal tracks but is less effective in country, jazz, and rock music.

These results infer that each feature space has its own advantage over a particular music genre. A rational expectation is that combining the three timbre attributes should create a better system than using a single feature space. This is validated in the next experiments.

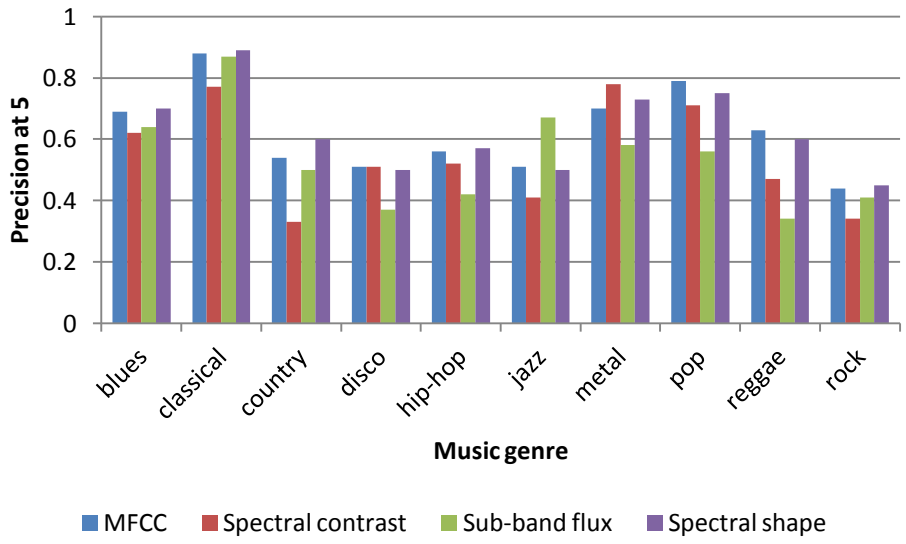


FIGURE 3.9: Precision at 5 per genre, per single feature space using the GTZAN dataset.

3.4.2 Combining features

A simple method of combining the features is to use a linear combination of the respective distance measures of each feature space. But combining the distances of each feature space is not as straightforward as it seems. The main issue is that each feature space has its own numerical range. We can solve this by normalizing or *standardizing* the distance values. Suppose that for a particular feature space, we have an NxN distance matrix, see Figure 3.10. For each distance measure d_{xy}, we collect all the distances at row x and column y. This forms a subset of distance measures. The reason for limiting the subspace is we are only concerned with the relative score of $d_{x,y}$ with respect to other distance measures for songs s_x and s_y. We solve for its mean $\mu_{x,y}$ and standard deviation $\sigma_{x,y}$. The normalized distance measure is solved as:

$$d'_{x,y} = \frac{d_{x,y} - \mu_{x,y}}{\sigma_{x,y}} \qquad 3.12$$

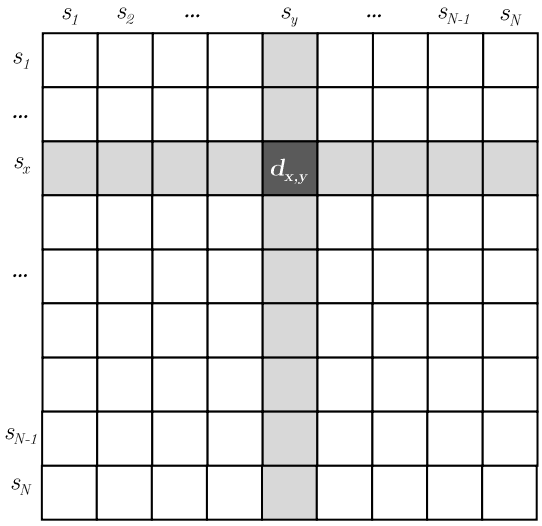


FIGURE 3.10: Normalization of a distance matrix.

In this way, the distance measures are transformed into standard scores that allow them to be combined with the scores from other feature spaces. Each distance measure between two songs s_x and s_y has its own normalization parameters since only the distances along row x and column y are used for normalization. Thus, this operation also affects the ordering within a row or column with respect to the original distance matrix. Figure 3.11 shows a histogram of the change in row ranks in a distance matrix. Based on the histogram, 93% of the distance values changed its rank or position within a row after normalization. The average change of position within a row is almost 70.

An unexpected effect of normalization is it improves the precision of the single feature spaces, refer to Table 3.14. The highest improvements are observed using the 1517-Artists dataset. The average precisions for the spectral contrast, sub-band flux, and spectral shape increased by 8.5%, 3.4%, and 8.2%, respectively.

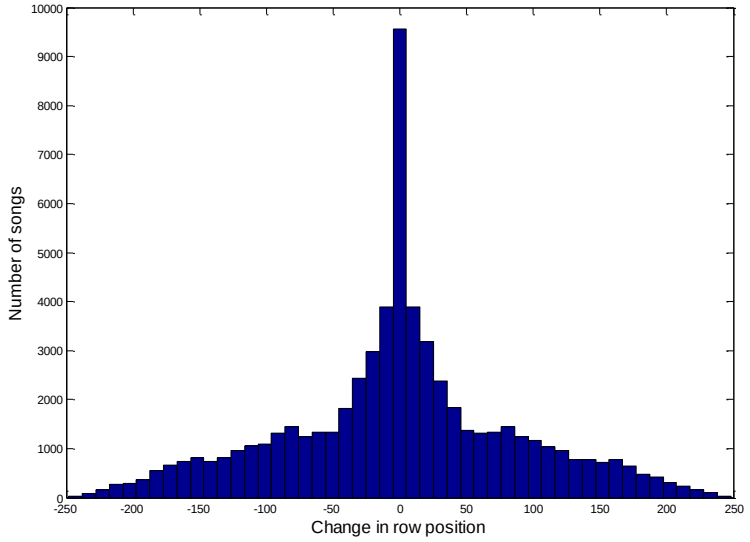


FIGURE 3.11: Histogram of the changes in row position due to normalization; evaluated on the DB-S dataset using the spectral shape features.

TABLE 3.14: Summary of precision before and after normalization.

Features	*normalized*	DB-S				GTZAN			
		P_5	P_{10}	P_{15}	P_{20}	P_5	P_{10}	P_{15}	P_{20}
MFCC (baseline)		0.79	0.75	0.72	0.69	0.63	0.56	0.52	0.48
Spectral contrast	no	0.77	0.72	0.69	0.66	0.54	0.49	0.45	0.42
	yes	0.78	0.74	0.7	0.68	0.58	0.52	0.49	0.46
Sub-band flux	no	0.71	0.66	0.61	0.58	0.54	0.48	0.44	0.41
	yes	0.72	0.66	0.62	0.6	0.55	0.49	0.45	0.42
Spectral shape	no	0.81	0.77	0.73	0.71	0.63	0.56	0.52	0.49
	yes	0.82	0.79	0.75	0.72	0.65	0.59	0.54	0.51
Combined		**0.86**	**0.83**	**0.80**	**0.76**	**0.71**	**0.65**	**0.6**	**0.56**
		1517-Artists				Unique			
MFCC (baseline)		0.40	0.36	0.34	0.33	0.39	0.37	0.36	0.35
Spectral contrast	no	0.36	0.33	0.31	0.30	0.37	0.35	0.34	0.33
	yes	0.38	0.36	0.34	0.33	0.37	0.36	0.36	0.35
Sub-band flux	no	0.32	0.30	0.28	0.28	0.33	0.32	0.31	0.31
	yes	0.32	0.31	0.30	0.29	0.33	0.32	0.32	0.32
Spectral shape	no	0.40	0.37	0.35	0.34	0.39	0.38	0.37	0.36
	yes	0.42	0.40	0.39	0.37	0.39	0.38	0.37	0.37
Combined		**0.46**	**0.43**	**0.41**	**0.39**	**0.42**	**0.41**	**0.40**	**0.39**

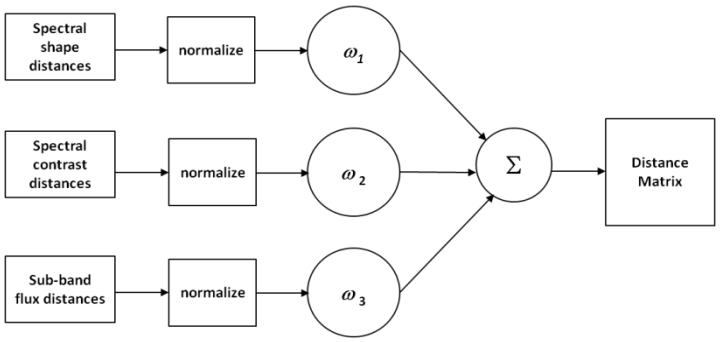


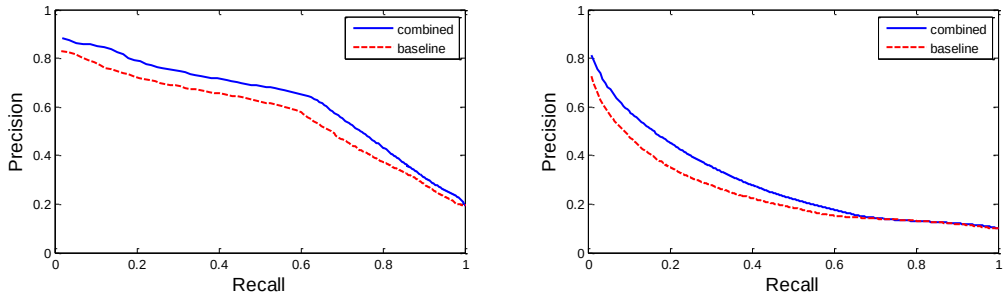
FIGURE 3.12: Block diagram on combing the distance measures.

After normalizing the distances for each feature space, the full distance matrix is computed as a weighted combination of the normalized scores, see Figure 3.12. The problem with this approach is that humans do not judge music similarity as a weighted sum of similarities with respect to different features. Nevertheless, this simple approach allows us to further optimize the system by determining the weights for each feature space that maximize precision.

To determine the optimum weights, ω_1, ω_2, and ω_3 are varied in the range of 0 to 1 with a step size of 0.1. The optimum weights are determined from the combination that results in the best precision at 20. The values in bold in Table 3.14 show the results using the optimum weights. The combined features clearly outperform any of the individual features. Comparing the improvements with the baseline, the average increase in precision ranges from 10.2% (DB-S) to 18.2% (1517-Artists). The only problem is the inconsistency of the weights to be used as each dataset has a different set of optimum weights. Table 3.15 lists the optimum weights. It shows however that each component has a positive effect on the performance since none of the features has a weight of zero. Thus, we can conclude that all the feature spaces contribute to the overall performance of the timbre similarity algorithm.

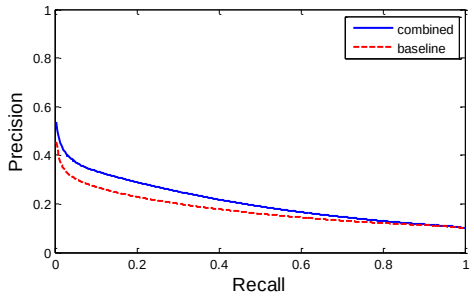
TABLE 3.15: Optimum weights for each feature

Features	DB-S	GTZAN	1517-Artists	Unique
Spectral contrast	0.8	1	0.8	0.6
Sub-band flux	0.3	0.2	0.2	0.4
Spectral shape	0.2	0.5	0.8	0.4

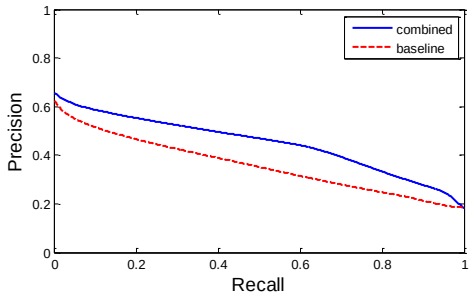


(a) DB-S

(b) GTZAN



(c) 1517-Artists



(d) Unique

FIGURE 3.13: Average Precision-Recall curves of the baseline and combined features system.

We can also compare the baseline system with the combined features algorithm by plotting their average precision-recall curves using leave-one-out cross validation, see Figure 3.13. The optimum weights are used for respective datasets. While this may be viewed as overfitting, we are more interested in comparing the baseline system with the combined features system. Hence, comparisons of performance among datasets are not made. The plots show that the combined features system (blue curve) consistently outperforms the baseline system (red curve). This means that for the same number of returned items, more relevant results are obtained using the combined features system than the baseline system.

From Table 3.16, we can see that the combined features system has significantly higher mean average precision than the baseline system. For example using the Unique dataset, the mean average precision improved by as much as 25%. This means that relevant results are returned earlier in the returned list. The mean average precision for DB-S is the highest among the datasets due to the presence of multiple songs from the same

artist. There is a higher probability that songs from the same artist to the query song are returned. This observation is also called artist effect.

TABLE 3.16: Comparison of mean average precision between the baseline system and the combined features system

Algorithm	DB-S	GTZAN	1517-Artists	Unique
baseline	0.63	0.29	0.21	0.39
combined	0.69	0.34	0.25	0.49

3.5 Hubs and orphans

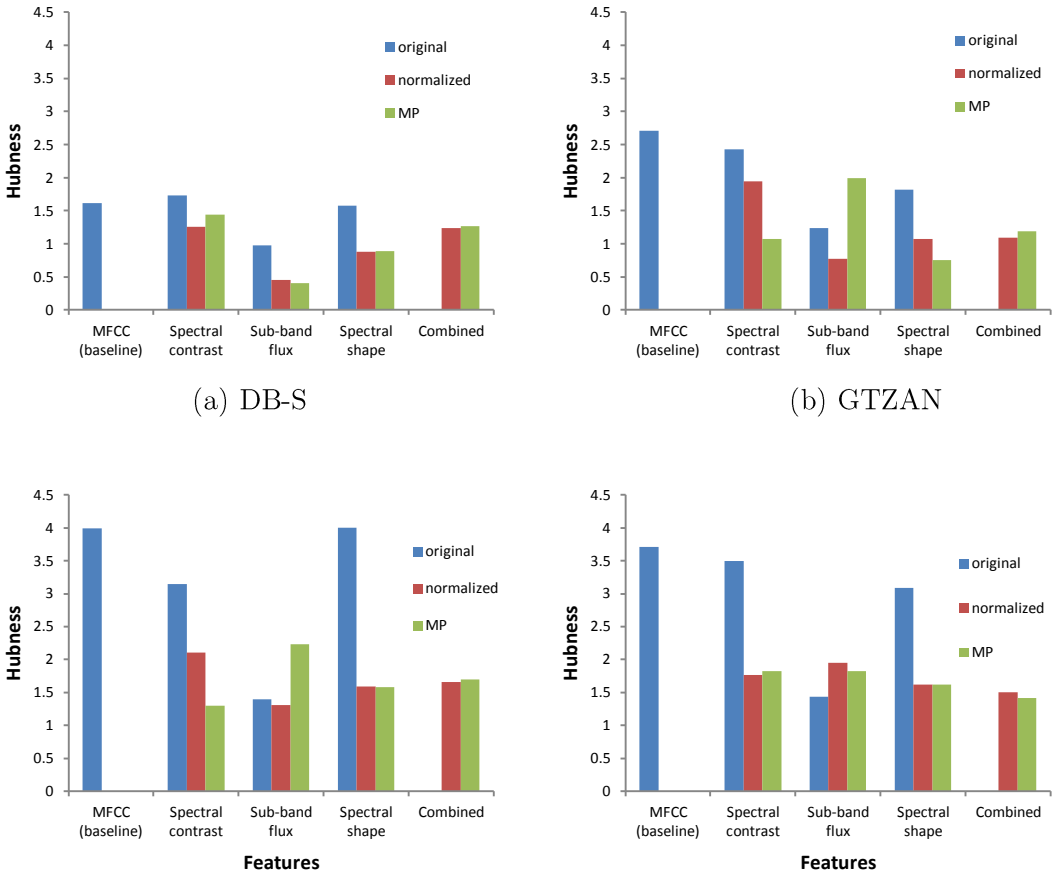
In Section 3.3.3.2, we introduced the concept of *hubness*. It is an undesired phenomenon and is considered a general problem of machine learning in high dimensional spaces (Karydis et al. 2010). Hubs are items that appear to be 'always similar' to a large number other items. This phenomenon severely hampers the performance of similarity search systems, as other related items have less probability of being retrieved.

There are a number of approaches that have been proposed to address this problem such as the non-iterative contextual dissimilarity measure (NICDM) (Jegou et al. 2010) and Mutual Proximity (Schnitzer et al. 2012). In this work, we will show that the normalization process applied to the feature spaces has an added benefit of reducing the hubs. We will also compare the results when Mutual Proximity (MP) is used on the distance matrices.

To evaluate the impact of normalization on the timbre similarity algorithms we look at their change in *hubness* (Radovanović et al. 2010). Let $N_k(i)$ be the number of times song i appears among the k nearest neighbours of all other items in a database, with respect to some similarity measure. The *hubness* S_{Nk} is measured as the skewness of the distribution of $N_k(i)$.

$$S_{N_k} = E(N_k - \mu_{N_k})^3 / \sigma_{N_k}^3 \qquad 3.13$$

where μ_{Nk} and σ_{Nk} are the mean and standard deviation of N_k respectively. Higher values of skewness indicate higher hubness.



(a) DB-S

(b) GTZAN

(c) 1517-Artists

(d) Unique

FIGURE 3.14: Hubness values decrease significantly after normalizing the distance matrices, k=5.

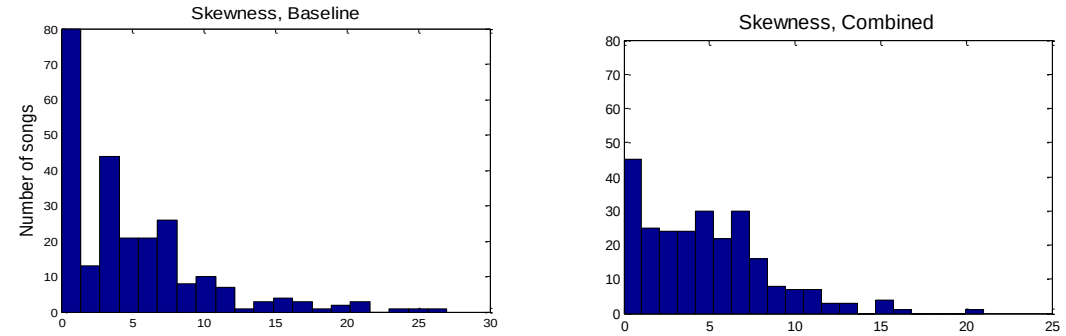
Figure 3.14 shows the hubness values for the different feature spaces used in this work. Without normalization (blue bars) the hubness values are high. It can be observed that the level of hubness is correlated with the size of the dataset. Using the baseline algorithm (20-dimensional), the minimum value is 1.6 (DB-S, 250 songs) while the maximum value is 4 (1517-Artists, 1702 songs). The hubness values using spectral contrast and spectral shape are sometimes comparable to the baseline algorithm. But the values for sub-band flux are considerably lower than the others even without normalization. A possible explanation for this is that the dimensionality of sub-band flux (10-dimensional) is much lower than other features.

If we use this logic regarding dimensionality, then the spectral shape features with dimensionality of 29 should always have the highest hubness values. However, this is not the case, particularly in the GTZAN and Unique datasets. We can conclude that while

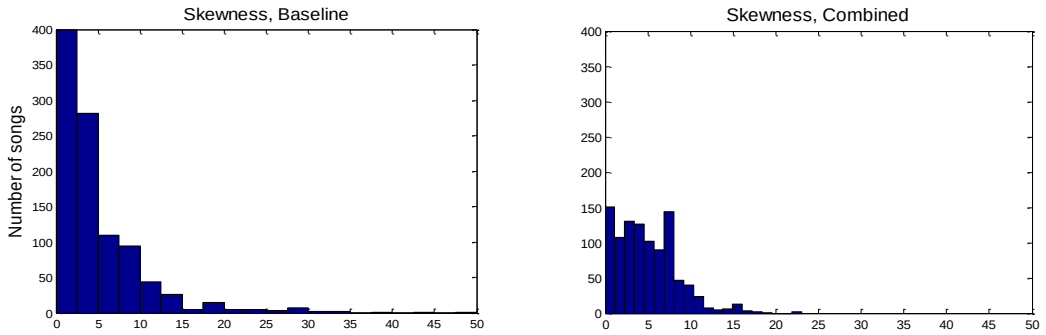
the baseline and spectral shape features have similar performance in terms of precision, the spectral shape features have an advantage of producing fewer hubs.

Applying normalization (red bars) and Mutual Proximity (green bars) on the feature spaces reduced the hubness except on some cases, e.g. sub-band flux using GTZAN, 1517-Artists, and Unique datasets. For the DB-S and Unique datasets, the effects of normalization and MP are similar. For the remaining datasets, normalization works much better for sub-band flux while MP is more effective for spectral contrast. When the feature spaces are aggregated, the resulting combined feature space has significantly lower hubness compared to the baseline algorithm. For the DB-S dataset, the reduction is almost 50% while for the other datasets the reduction is around 60%.

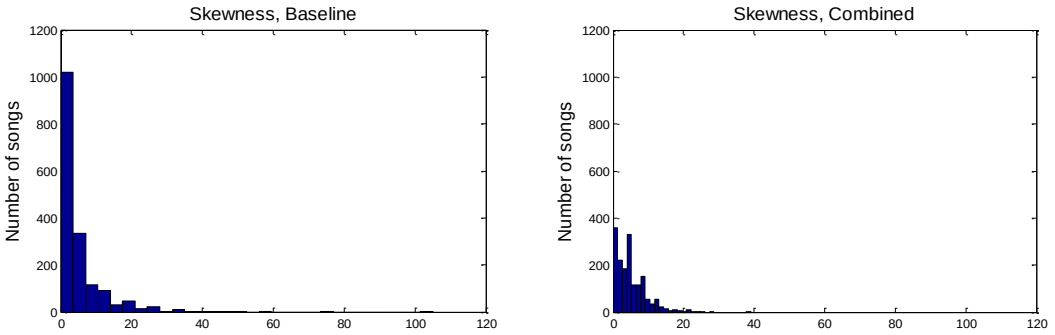
Figure 3.15 compares the histograms of the $k = 5$ occurrences of the baseline algorithm and the combined features algorithm. The figures show us how often the songs in a database appear in the top 5 list. Ideally, we want a symmetric distribution concentrated around 5. The first column of the figure shows the histograms without normalization. In all cases, we can see that the distributions are highly skewed. The high number of songs in the lower range is offset by some songs in the higher range. The songs in the higher range are 'always similar' songs or *hubs* that decrease the probability of the relevant songs being retrieved. This is evidenced by the very high values in the first bin in the histogram. The songs in this bin are the orphans. Applying normalization and combining the feature spaces visibly reduces the number of orphans and hubs. The performance of our methods in reducing hubs is comparable with the state-of-the-art, this will be shown in Section 4.5.3.



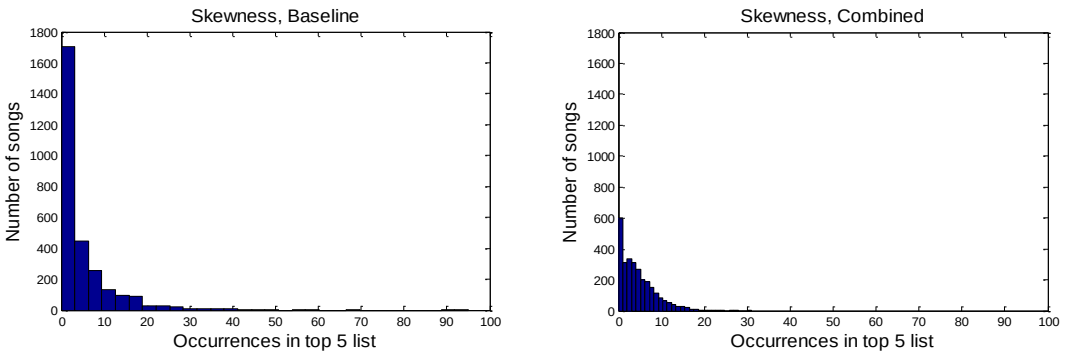
(a) DB-S



(b) GTZAN



(c) 1517-Artists



(d) Unique

FIGURE 3.15: Comparing the histograms of the $k = 5$-occurrences of baseline system (first column) with the combined features system where normalization is applied (second column). The distribution of the occurrences is more uniform for the combined features.

TABLE 3.17: Percentage of orphaned songs in the different databases in the top 5 lists.

Algorithm	DB-S	GTZAN	1517-Artists	Unique
MFCC (baseline)	18.0%	15.5%	24.6%	25.3%
Combined	11.6%	4.7%	9.0%	9.1%

Table 3.17 compares the percentage of orphaned songs of each database for both baseline MFCC and combined features algorithms. In the 1517-Artists and Unique datasets, almost a quarter of the songs will never be returned in the top 5 lists. With the combined features, the number of orphans is reduced to just 9%.

The computational complexity of applying normalization is $O(N^2)$. This is problematic for commercial applications that contain millions of tracks. A possible solution is to limit the search space to which the normalization process will be applied. This will be explored in Section 4.3.2.

3.6 Summary

This chapter described the processes for estimating timbre similarity. It begins with audio pre-processing where the signal is normalized and segmented into frames. Given that the segmented signal is pseudo-stationary, the Fourier transform can be applied to convert the time domain into frequency domain. At this point, several signal processing techniques are applied to derive timbre features. In this work, we extract Mel-frequency cepstral coefficients, Mel-based spectral contrast coefficients, sub-band flux coefficients, and nine spectral distribution descriptors. Each of these features is motivated by the different attributes and perceptual correlates of timbre. The single multivariate Gaussian distribution is selected to model the features and a transformed Kullback-Leibler divergence for single multivariate Gaussian distributions is used for similarity estimation.

Feature parameters, model parameters and dynamics of the data in optimizing our system are considered. Specifically, the dimensionality of MFCCs, modified spectral contrast, and sub-band flux are varied. The delta, delta-delta coefficients, and spectral distribution descriptors are appended with MFCCs to test if the spectral envelope feature

space can be improved. Finally, the features spaces are combined to create a multi-dimensional polyphonic timbre space.

The results of evaluating different timbre features are presented. To avoid the problems of overfitting, we ran the experiments using four datasets. The preliminary experiments were done using GTZAN and the author's DB-S dataset. By varying the significant parameters of a feature space, we determined the optimum values that maximize the performance. Once the general parameters were determined, further evaluations were done using 1517-Artists and Unique datasets. This enabled us to validate our results and to make conclusions regarding their general applicability across the different datasets.

Four feature spaces were evaluated: 1) spectral envelope represented by MFCCs, 2) spectral contrast, 3) sub-band flux, and 4) spectral distribution descriptors. The performance of the MFCCs served as the baseline. We evaluated MFCCs appended with its time derivatives but no significant improvements were observed. In a similar manner, we tried appending the MFCCs with the spectral distribution descriptors. This resulted in a slight improvement in terms of precision and we called this new feature space *spectral shape*. The precision results from each of the three feature spaces indicate that each feature space works better in some genres.

We proposed a framework of combining the feature spaces by normalizing the distance values from a particular distance matrix. Linearly combining the feature spaces resulted in the increase of precisions compare to the results using any individual feature space. Moreover, normalizing the distance matrices has an additional benefit of reducing the presence of hub and orphan songs.

Chapter 4: Towards Efficient Similarity Estimation

We have developed two blocks of a content-based audio retrieval system: 1) feature extraction, and 2) similarity computation. In this chapter, we tackle the problem of scalability to find solutions that can be easily integrated into our system. The result of our investigations is a candidate audio retrieval system that is both effective and efficient. We validate our candidate system by submitting our algorithms to MIREX 2013. The MIREX results show that the performance of our candidate system is comparable with state-of-the-art systems.

4.1 Introduction

At present the largest digital music store, iTunes, contains over 37 million tracks[32]. Despite technological advances, content-based MIR is not integrated with iTunes. Next generation MIR systems must address this and bring content-based methods to the larger music services and digital libraries.

As an illustration, the MIREX audio similarity experiment requires computation of 7000 x 7000 similarity matrices, so the task involves pair-wise track comparisons on the order of millions. However, today's music download services and music and video downloads sites are on the order of tens of millions of tracks, so pair-wise comparisons require computation on the order of hundreds of trillions. Clearly, pair-wise methods are difficult at this scale on even the most advanced hardware; however, this scale is required to make content-based MIR applicable to solving today's media search and retrieval problems.

[32] http://www.apple.com/itunes/what-is/

Furthermore, most of the similarity estimation techniques that have been demonstrated in literature have been based on the application of distance measure to features computed from audio tracks in the dataset, leading to a linear scaling of computation time with dataset size (given sufficient resources). Hence, a system effectively returning results in 100 ms on a 10,000-track dataset might require a full 10 seconds to conduct a search of 1,000,000 tracks. If we are to provide efficient music search techniques, either much faster distance computations or more efficient indexing strategies that scale sub-linearly with dataset size will have to be found.

4.2 Survey

The most important issue with indexing high-dimensional spaces is how to reduce the amount of data searched. If this can be done successfully, this has the benefits of reducing 1) the number of distance computations and 2) the amount of data loaded from disk or memory. The following sections present background information on attempts to solve this problem.

4.2.1 Multidimensional scaling

Multidimensional scaling (MDS) aims to find a low-dimensional projection of a data to preserve, as closely as possible, the pair-wise distances between data points (Bishop 2007). This method requires 1) a set of N objects, 2) their pair-wise similarities and 3) the desired dimensionality k. The algorithm will map each object to a point in the k-dimensional space while minimizing a loss function called *stress*.

To achieve this, MDS begins with an estimate and iteratively improves it, until no further improvement is possible. The algorithm works roughly as follows: It originally assigns each object to a k-dimensional point. It then computes the distance from the other N-1 points and moves the point to minimize the difference between the actual similarities and the estimated k-dimensional distances. The pair-wise distances are analogueous to 'springs'; then the algorithm tries to reorganize the positions of the k-dimensional points to minimize the stress of the springs.

MDS has been used in numerous, diverse applications such as analysis of coded speech (Hall 2013), biomedical engineering (Meyer and Wolf 1997), personality traits relationship (Wu and Cao 2011), localization through wireless sensor network (Cabero et al. 2008), etc. However, for content-based retrieval systems, MDS has the following disadvantages:

1. The computational complexity of MDS is $O(N^2)$. Hence, it is impractical for large datasets.
2. In the 'query-by-example' application, the query object has to be mapped to a point in k-dimensional space. The complexity of searching/adding a new item in the database would be the same as sequential scanning since the pair-wise distances need to be computed.

4.2.2 Dimensionality reduction

In MDS, the objective is to map objects in a k-dimensional space given the distances between them. Another scenario is when we have already extracted features from the objects, and we want to map them to a lower dimensional space. This problem has been studied extensively in statistical pattern recognition and matrix algebra. The optimal way to map n-dimensional points to k-dimensional points ($k \leq n$) is the Karhunen-Loéve (K-L) transform (Bishop 2007). It is optimal in the sense that it minimizes the mean square error, where the error is the distance between each n-dimensional point and its k-dimensional image. This technique is also known as Principal Component Analysis (PCA).

PCA is often used in pattern matching to choose the most important dimensions for a given set of vectors. It computes the eigenvectors of the covariance matrix, sorts them in decreasing eigenvalue order, and approximates each data vector with its projections on the first k eigenvectors. The process is closely related to Singular Value Decomposition (SVD) (Shlens 2009).

The following assumptions are made behind PCA that lead to its limitations:

1. It assumes linearity as data are projected linearly onto a subspace of lower dimensionality than the original space. Hence, PCA will work well if the dimensions are linearly related to each other.

2. It assumes that the data has a high signal-to-noise ratio. Principal components with larger associated variances represent important structures, whereas lower variances represent noise. However, higher-order dependencies might occur in the data and the resulting lower variances does not necessarily represent noise.

The main motivation behind this method is to decorrelate the data. If the original data were uncorrelated, PCA does nothing but sort them according to their variance. In our case, the application of discrete cosine transform on the feature vectors already decorrelated the dimensions. Hence, performing PCA on the features will just be a redundant operation.

4.2.3 Spatial access methods

If the similarity between data is represented by a distance metric, then spatial access methods can be used for quick retrieval. The main purpose of spatial access methods is to support efficient spatial selection, for example nearest neighbour queries of objects (Longley et al. 1999). The most popular methods fall under three classes: 1) tree-based methods like R-tree (Guttman 1984), and its variants {R^+-tree (Sellis et al. 1987), Hilbert R-Tree (Kamel and Faloutsos 1993), R*-tree (Beckmann et al. 1990), sphere tree, etc}; 2) methods using quadtree (Gargantini 1982), space filling curves like Z-order (Orenstein and Manola 1988); and 3) methods that use grid-files (Hinrichs 1985).

The timbre similarity methods used in this work are non-vectorial (i.e. features used are represented by single multivariate Gaussian models) and use non-metric divergence measures (e.g. Kullback-Leibler divergence). Hence, the spatial access methods cannot be directly used.

4.2.4 FastMap

The FastMap algorithm (Faloutsos and Lin 1995) is a promising approach for fast searching since it only requires a metric distance function to embed the features in a k-dimensional Euclidean space. This will enable the use of spatial access methods for quick retrieval. The application of FastMap for audio retrieval and browsing was described in

(Cano et al. 2002). They used MDS and FastMap to map collections of commercial songs and isolated instrument sounds to a 2-d space for visualization. They reported that MDS maps better than FastMap at the expense of high computational cost. Using FastMap, they achieved better results with isolated instrument sounds than commercial songs. They argued that it is harder to design good similarity measures for commercial songs.

We have demonstrated in Chapter 3 that our system provides a better estimate of polyphonic timbre similarity than using standard MFCCs. Given that the system uses approximated metric distance functions, we can also apply FastMap. We alter the original FastMap algorithm by taking advantage of insights gained from our experiments.

The main idea of FastMap is to project N objects on a carefully selected imaginary line. This requires choosing two objects O_a and O_b, which will be called *pivot objects*, and consider the line that passes through them on n-d space. To determine the pivot objects, we choose an arbitrary object and let it be the second pivot object O_b. Then, compute the distance to all other objects using the symmetrised Kullback-Leibler divergence (SKL). Next, transform the distances into metric using Equation 2.28. After which, sort the distances and select the median object, instead of the farthest object as proposed in the original algorithm, as the first pivot object O_a. The reason for selecting the median is that it decreases the probability of selecting an *orphan* (always dissimilar) object as the normalization process that reduces hubs and orphans is not yet applied at this point. Lastly, update the second pivot object O_b by selecting the median object after computing all the distances from the first pivot object O_a.

Once the two pivot objects are selected, the projections of the objects on the line that passes through them are computed using the cosine law, see Figure 4.1. Using the cosine law, the projection of an object O_i can be solved in terms of the distances between the pivot objects:

$$d_{b,i}^2 = d_{a,i}^2 + d_{a,b}^2 - 2d_{a,b}d_{a,i}\cos\alpha \Rightarrow d_{a,i}\cos\alpha = x_i = \frac{d_{a,i}^2 + d_{a,b}^2 - d_{b,i}^2}{2d_{a,b}} \quad 4.1$$

Using Equation 4.1, we can map all the objects in a database into points on a line while preserving some of the distance information. For example, if O_i is very close to O_a, then x_i will be small. The projections represent the embedding of the objects in the 1st dimension.

This approach can be extended so that the objects can be embedded into points in 2-dimensional space. Consider a hyper-plane H that is perpendicular to the line (O_a, O_b), see Figure 4.2 where the objects are projected on the hyper-plane. Let O_i' and O_j' represent the projections of objects O_i and O_j on the hyper-plane. If we can solve for the distance between the two projected objects, then we can apply the same methods that were used in embedding the objects in the 1st dimension.

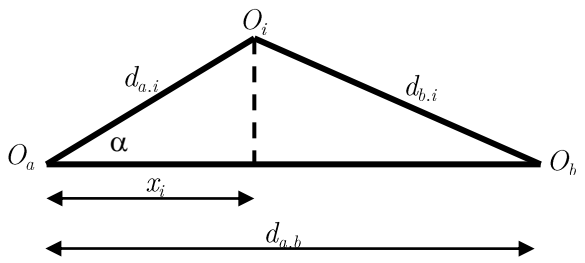


FIGURE 4.1: Visualizing the cosine law. The lengths of the sides $d_{a,i}$ and $d_{b,i}$ are the distances of the object O_i to the two pivot objects, the length $d_{a,b}$ is the distance between the two pivot objects. The length x_i can be computed using Equation 4.1.

Refer to Figure 4.2. Using Pythagorean theorem on triangle $O_i C O_j$,

$$(CO_j)^2 = (O_i'O_j')^2 = O_iO_j^2 - (x_i - x_j)^2 \quad i, j = 1 \ldots N \tag{4.2}$$

We can solve for x_i and x_j using Equation 4.1. The distance between O_i and O_j can be computed using SKL. Thus, we can compute the distance between the projections O_i' and O_j' using Equation 4.2. A new pair of pivot objects is selected and the projections are computed using Equations 4.1 and 4.2. The FastMap algorithm is run recursively until the target dimension is reached. The output is an N x k projection matrix where the i^{th} row is the image of the i^{th} object.

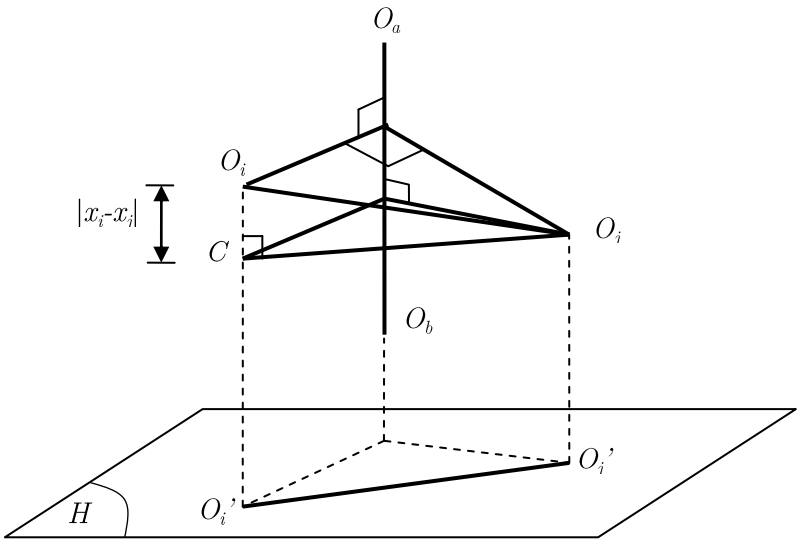


FIGURE 4.2: Projecting objects on a hyper-plane H, perpendicular to the line O_aO_b.

4.3 Using FastMap for timbre similarity estimation

Now that we are able to map the features into Euclidean space, the process of similarity computation can be done using Euclidean distance instead of the more computationally intensive Kullback-Leibler divergence. A single Euclidean distance calculation, for an n dimensional feature, will have n subtractions, n multiplications and $(n - 1)$ additions for total of $(3n\text{-}1)$ floating-point operations (flops). Meanwhile, the symmetrised KL (SKL) divergence requires 4 matrix multiplications[33], 3 matrix traces[34] and 3 additions for total of $(8n^3 - 4n^2 + 6n)$ floating point operations. Hence, the computational complexity of Euclidean distance and KL divergence is $O(n)$ and $O(n^3)$ respectively. Clearly, SKL is more expensive to compute than Euclidean distance. If the performance of the system using FastMap is as good as the original system for timbre similarity estimation, then it is a good candidate for commercial applications since the spatial access methods discussed in Section 4.2.3 can be used.

[33] matrix multiplication requires $n*n(2n\text{-}1)$ flops
[34] matrix trace requires $2n\text{-}1$ flops

4.3.1 Preliminary experiments

The schematic for the retrieval system using FastMap is shown in Figure 4.3. The first step is to embed all the features into the Euclidean space using FastMap. After this, similarity computation can be computed using Euclidean distance to return a set of nearest songs to a query. For the preliminary experiments, we evaluated the system using the modified FastMap for each of the feature space. The objectives are: 1) to determine the effects of the number of the target mapping dimensions k on the precisions, and 2) to compare the performance of using Euclidean distance against using KL divergence.

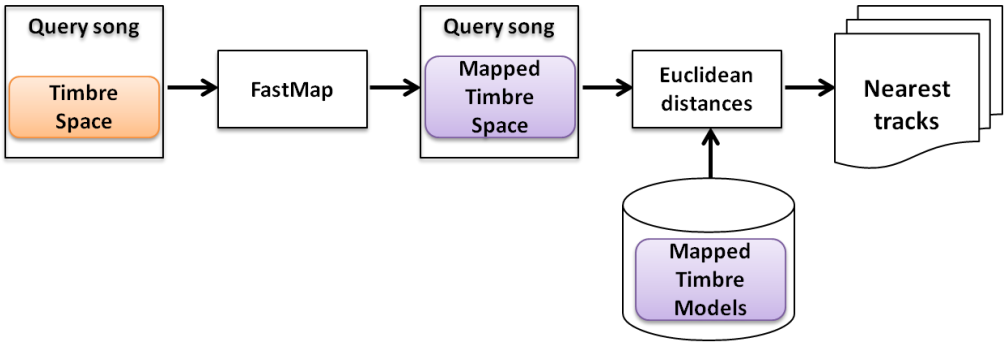
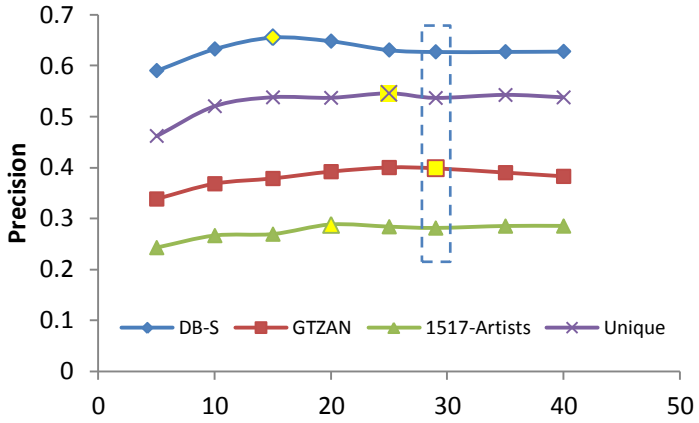


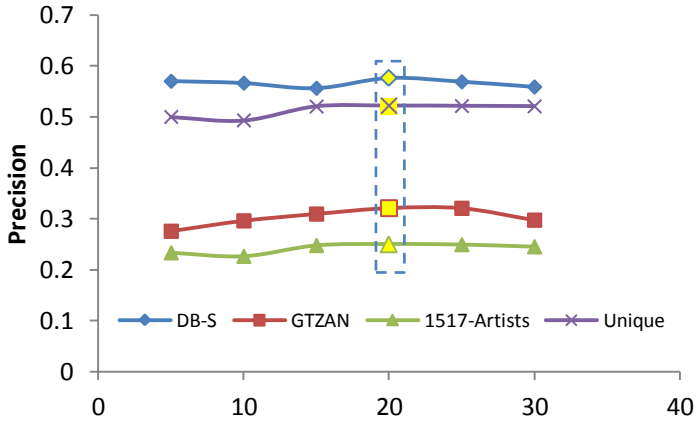
FIGURE 4.3: Block diagram of the retrieval system using FastMap.

Figure 4.4 shows the effects of varying the target dimension k on the performance of the feature spaces. The precision using the original dimension of the feature space is boxed in dashed line. Our initial assumption is that the precision increases as k increases. This trend is most apparent in the *spectral shape* for all datasets except DB-S. An interesting result is that it is possible to achieve better performance at a lower value of k than the original. For example, the highest precision for the spectral shape feature space occurs when $k = 15$ using the DB-S dataset. This is almost half of its original dimension of 29. This is also observed for the *sub-band flux* where the best precision occurs when $k = 8$ using GTZAN and 1517-Artists datasets. The highest values of precision are highlighted for values of k less than or equal to the original dimensionality.

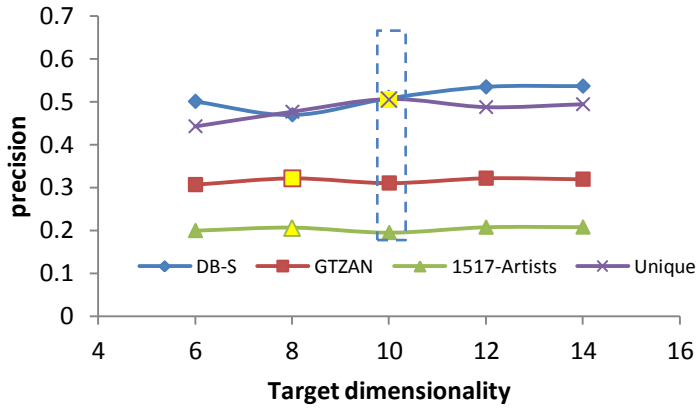
We also evaluated the case where k is greater than the dimensionality of the original space. In almost all cases, no further significant improvements are observed except for the sub-band flux feature using DB-S dataset.



(a) Spectral shape



(b) Spectral contrast



(c) Sub-band flux

FIGURE 4.4: Precision at 20 as a function of the target space dimension k for each feature space: (a) spectral shape, (b) spectral contrast, and (c) sub-band flux.

We next compare the performance of the system using Euclidean distance against using symmetrised KL divergence (SKL) to compute timbre similarity. We used the optimum value of k for each feature space based on the previous evaluations. The results are listed in Table 4.1. It shows that all the precision values decreased in comparison with the precision values obtained using SKL shown in Table 3.14. The variance in the decrease in precision is small within a dataset but varies widely among the four datasets. The most significant decrease in performance is observed using the GTZAN dataset where the precision values decreased by an average of 0.15; for the DB-S, 1517-Artists, and Unique datasets, the average decrease is 0.09, 0.08, and 0.05, respectively.

TABLE 4.1: Summary of retrieval performance for different feature spaces after applying modified FastMap.

Dataset	*Features*	*Euclidean distance*				*Decrease in precision relative to SKL*			
		P_5	P_{10}	P_{15}	P_{20}	P_5	P_{10}	P_{15}	P_{20}
DB-S	Spectral contrast	0.66	0.63	0.60	0.59	0.13	0.11	0.10	0.09
	Sub-band flux	0.63	0.59	0.54	0.51	0.09	0.08	0.08	0.09
	Spectral shape	0.73	0.70	0.67	0.66	0.09	0.09	0.08	0.07
	Combined	0.82	0.76	0.71	0.68	0.05	0.08	0.09	0.09
GTZAN	Spectral contrast	0.39	0.36	0.34	0.32	0.19	0.16	0.15	0.14
	Sub-band flux	0.39	0.35	0.33	0.31	0.16	0.14	0.12	0.11
	Spectral shape	0.48	0.43	0.41	0.39	0.17	0.15	0.13	0.13
	Combined	0.56	0.50	0.46	0.44	0.16	0.14	0.14	0.12
1517-Artists	Spectral contrast	0.27	0.26	0.25	0.24	0.11	0.08	0.08	0.07
	Sub-band flux	0.22	0.21	0.20	0.20	0.10	0.09	0.08	0.07
	Spectral shape	0.33	0.31	0.30	0.30	0.09	0.08	0.07	0.06
	Combined	0.35	0.33	0.32	0.31	0.11	0.10	0.09	0.08
Unique	Spectral contrast	0.32	0.30	0.30	0.30	0.06	0.06	0.06	0.05
	Sub-band flux	0.27	0.27	0.26	0.26	0.06	0.06	0.06	0.06
	Spectral shape	0.34	0.33	0.32	0.32	0.05	0.05	0.05	0.05
	Combined	0.35	0.34	0.34	0.33	0.07	0.06	0.06	0.06

Metric distance function

If the distance function used to estimate similarity is a *metric*, then the distance function is compatible with many algorithms and indexing methods that require metric properties, e.g. Locality-Sensitive Hashing (Datar et al. 2004), M-tree (Ciaccia et al. 1997), and PM-Tree (Skopal et al. 2004).

A metric distance function, D, is a scale that assigns to every pair of points a nonnegative number, called their distance, in accordance with the following three axioms (Tversky 1977):

Minimality:

$$D(a,b) \geq D(a,a) = 0$$

Symmetry:

$$D(a,b) = D(b,a)$$

Triangle inequality:

$$D(a,b) + D(b,c) \geq D(a,c)$$

FastMap embeds the objects in a Euclidean space. Hence, all the properties of a metric distance hold true, e.g. non-negativity, symmetry, and triangle inequality. After computing the Euclidean distances between the query object and the objects in the database for each feature space, the distances can be added linearly.

$$\omega_1 D_E(a,c)+\omega_2 D_E(a,c)+\omega_3 D_E(a,c) \leq \omega_1(D_E(a,b)+D_E(b,c))+\omega_2(D_E(a,b)+D_E(b,c)) + \omega_3(D_E(a,b)+D_E(b,c))$$

where ω is the optimum weight we have determined in Section 3.4.2 with respect to a particular feature space and $D_E(\ )$ is the Euclidean distance. The results of using the combined distances are shown in bold in Table 4.1. It can be observed that the precisions using the combined distances are consistently higher than using a single a feature space. This highlights the importance of each feature space on the estimation of timbre similarity. However, the values obtained are still lower compared to using exact

divergence. This means that after mapping the Gaussian models to the Euclidean space, some information regarding the proximities among the music tracks are not preserved.

A possible reason for the decrease in precisions is that FastMap assumes that the original objects exist in Euclidean space. A key aspect of FastMap is unique to Euclidean spaces, e.g. applicability of the Pythagorean theorem. In our case, the objects exist as Gaussian models, not points in Euclidean space. The modified FastMap algorithm uses the transformed KL divergence in projecting the objects in a hyper-plane. However, the transformation of KL divergence to metric distance is just an approximation. Thus, the computed distances between objects may not satisfy the Pythagorean theorem.

Another reason for the decrease in precisions can be attributed to the pivot object selection process. Instead of selecting the most distant object from a reference as a pivot object, we choose the median object to avoid selecting an *orphan*. Once the two pivot objects are determined, an imaginary line is draw through them where other objects are projected. Consider the scenario shown in Figure 4.5. In the figure, O_a and O_b are the pivot objects and $d_{a,b}$ is the distance between them. Because of our modification in the pivot selection process, it is possible that an object (O_i) exists whose distance from a pivot object is greater than the distance between the pivot objects, e.g. $d_{b,i} > d_{a,b}$. Using Equation 4.1, the resulting value of the projection (x_i) is negative. If the original algorithm is followed, the value of the projection is always positive.

Table 4.2 lists the precision values when the original FastMap implementation is used for embedding the features. For the DB-S dataset, the precision values are lower than using the modified FastMap. For GTZAN, 1517-Artists, and Unique datasets, the precision values are also lower except for the sub-band flux feature space. This implies that FastMap may not be effective for feature spaces with low dimensionality. Using combined feature spaces, the precisions are lower for DB-S and 1517-Artists while there are almost no differences for GTZAN and Unique datasets.

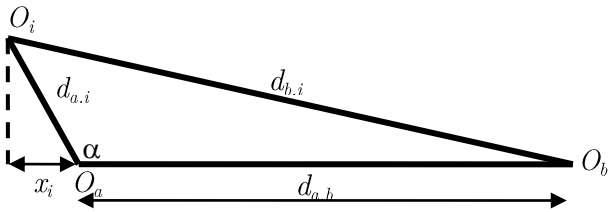


FIGURE 4.5: Visualizing a consequence of the pivot objects selection. The lengths of the sides $d_{a,i}$ and $d_{b,i}$ are the distances of the object O_i to the two pivot objects, the length $d_{a,b}$ is the distance between the two pivot objects. Since $d_{b,i} > d_{a,b}$, the value of the projection x_i is negative using Equation 4.1.

TABLE 4.2: Summary of retrieval performance for different feature spaces after applying original FastMap.

Dataset	*Features*	*Euclidean distance*				*Decrease in precision relative to Modified FastMap*			
		P_5	P_{10}	P_{15}	P_{20}	P_5	P_{10}	P_{15}	P_{20}
DB-S	Spectral contrast	0.65	0.62	0.6	0.58	0.01	0.01	0	0.01
	Sub-band flux	0.6	0.55	0.51	0.49	0.03	0.04	0.03	0.02
	Spectral shape	0.7	0.66	0.64	0.62	0.03	0.04	0.03	0.04
	Combined	0.78	0.73	0.7	0.68	0.04	0.03	0.01	0
GTZAN	Spectral contrast	0.38	0.35	0.33	0.32	0.01	0.01	0.01	0
	Sub-band flux	0.42	0.38	0.35	0.33	-0.03	-0.03	-0.02	-0.02
	Spectral shape	0.43	0.4	0.38	0.36	0.05	0.03	0.03	0.03
	Combined	0.56	0.51	0.47	0.44	0	-0.01	-0.01	0
1517-Artists	Spectral contrast	0.25	0.24	0.23	0.23	0.02	0.02	0.02	0.01
	Sub-band flux	0.22	0.22	0.21	0.21	0	-0.01	-0.01	-0.01
	Spectral shape	0.28	0.27	0.27	0.26	0.05	0.04	0.03	0.04
	Combined	0.33	0.31	0.3	0.29	0.02	0.02	0.02	0.02
Unique	Spectral contrast	0.29	0.27	0.26	0.26	0.03	0.03	0.04	0.04
	Sub-band flux	0.27	0.27	0.26	0.26	0	0	0	0
	Spectral shape	0.32	0.31	0.3	0.3	0.02	0.02	0.02	0.02
	Combined	0.36	0.34	0.34	0.33	-0.01	0	0	0

The approximation of the metric distance function and the modification of the pivot objects selection process contribute to violation of the *contractive* property of embedding, i.e. the distance between the mapped objects should be less than or equal to their original distance (Hjaltason and Samet 2003). The consistent decrease in precision as shown in Table 4.1 confirms that the contractive property did not hold for our adaptation of the FastMap algorithm.

To visualize the effect of mapping the objects using FastMap, we applied PCA on each mapped feature space to reduce it to a single dimension. The resulting mapped timbre space using the DB-S dataset is shown in Figure 4.6. If we compare it to Figure 3.7, we can say that the distribution of the objects is quite similar. The main difference is the clustering of music from the same genre is not as obvious as before. Based on the figure, only the hip-hop (cyan triangles) and classical genre (red squares) have clear clustering. The pop rock, electronic, and hard rock genre occupy almost the same space making it hard to distinguish between groups. This is reflected in the decrease in precision as listed in Table 4.1.

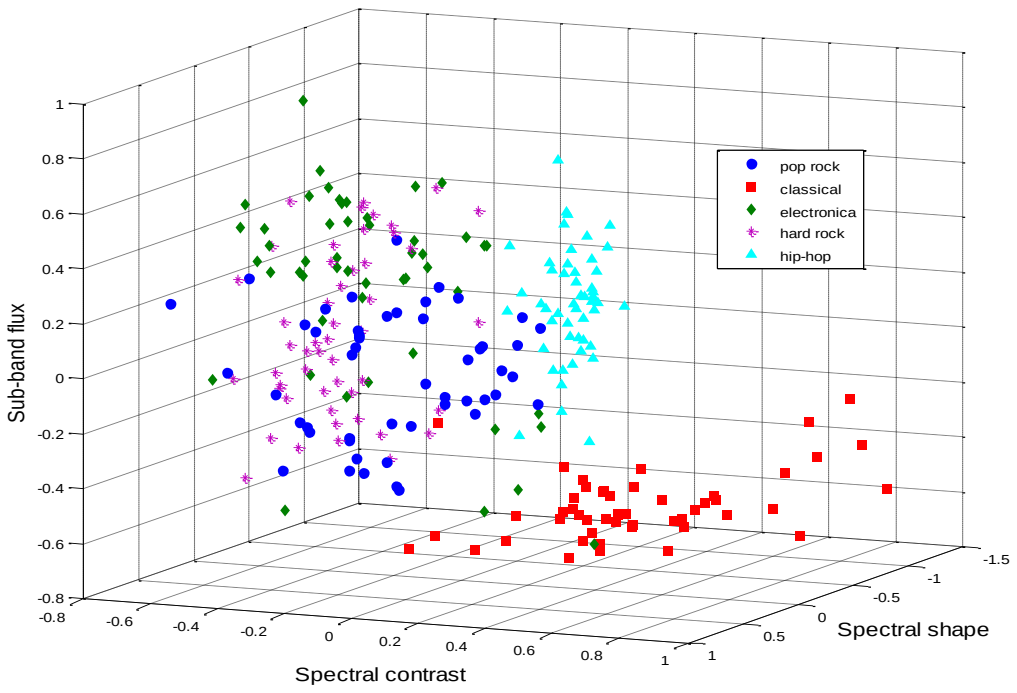


Figure 4.6: Mapping of the 250 DB-S music clips into the three-dimensional polyphonic timbre space after applying FastMap.

4.3.2 Filter and refine strategy

In the previous section, we have seen that mapping the objects to Euclidean space using FastMap has its advantages and disadvantages. Its main advantage is that similarity search can be performed much faster since Euclidean distance or efficient spatial indexing methods can be used. Its primary disadvantage is that the proximities between objects are not preserved. An alternative solution to speed-up similarity queries while achieving similar performance to the original system is to employ the *filter and refine* strategy (Jack 1989, Seidl and Kriegel 1998, G. R. Hjaltason and Samet 2003). It involves the following steps:

1. Filter step: The spatial index is used to rapidly eliminate the objects that could not satisfy the query. The result of this step is a set of candidates which includes relevant objects and some false hits.

2. Refinement step: Perform exact evaluation of candidates. Sort the results to obtain the nearest neighbours.

In the filter step, it is not necessary to reduce the objects to only those that are relevant to the query, but rather to remove a significant number of irrelevant objects in a computationally efficient manner. Hence, there is a trade-off between computation in the refine step and the time savings in the filter step. This approach has been successfully applied in multimedia applications such as browsing medical images (Korn et al. 1996) and audio retrieval (Schnitzer et al. 2009).

The filter and refine strategy can readily be adapted to our system, see Figure 4.7. Given a query song, its three feature spaces are mapped using FastMap. The Euclidean distance with respect to each feature space (spectral envelope, spectral contrast, sub-band flux) filters the music collection to return a number of possible nearest neighbours. The nearest neighbours form the candidate set. We will describe two approaches for the filtering step, *static* and *dynamic* filtering. The results are refined by computing the divergences on the aggregate candidate set with respect to the three feature spaces. The resulting distance matrices from each feature space are normalized and summed to produce the combined

distance matrix as described in Section 3.4.2. Lastly, the desired number of nearest objects is selected.

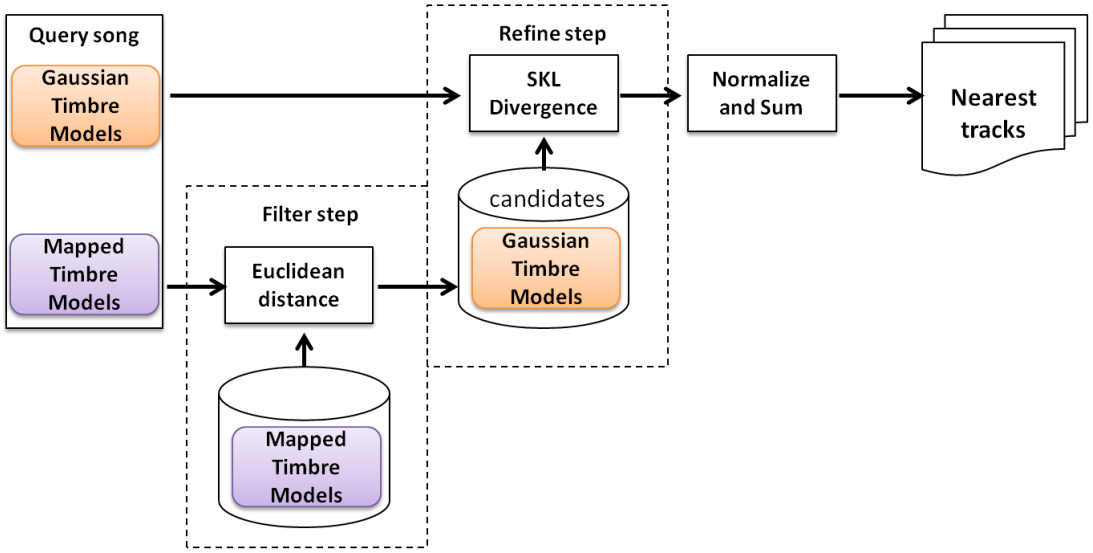


FIGURE 4.7: Block diagram of the proposed timbre similarity retrieval system using filter-and-refine strategy

4.3.2.1 Filter Step

The reliability of the filter step is crucial to the performance of the filter-and-refine strategy. The main parameter in the filter step is the number of possible nearest neighbours returned, called the *filter size*. To increase the reliability of the filter step, we can adjust the filter size relative to the desired number of nearest objects. It is expected that increasing the filter size will directly result in an improvement in the precisions. This is because there is a greater possibility that more relevant songs will be included to the candidate set relative to the target number of nearest neighbours. The main drawback is that the computational complexity also increases as more exact divergences are computed in the refine step. We describe two approaches in the filtering step: static and dynamic filtering.

Static filtering

It has been proven in Table 4.1 that combining the Euclidean distances from the feature spaces increases the precision. We adopt this approach in the filtering step, see Figure 4.8. The Euclidean distances of the feature spaces are combined linearly. The results are then sorted and the k-nearest neighbours (k = filter size) are returned as the candidate set. The process is *static* since the number of candidates returned by this approach is constant.

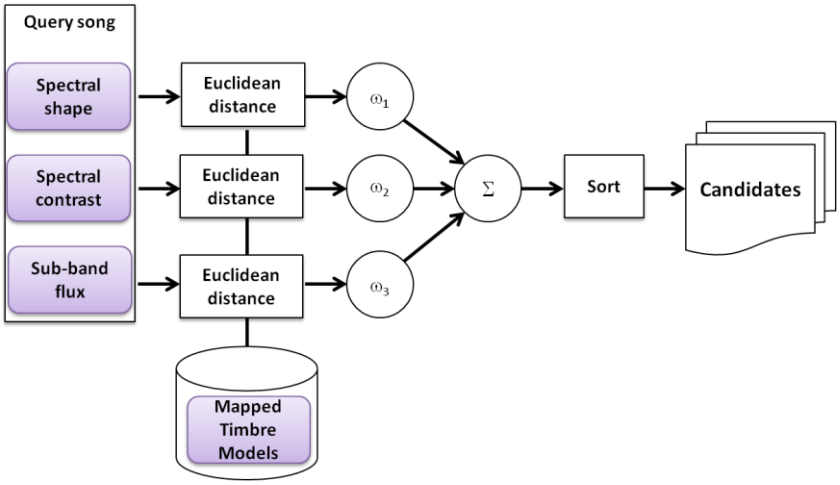


FIGURE 4.8: Block diagram of the filter step using static filtering.

Dynamic filtering

Combining the Euclidean distances from the feature spaces increases the precision than using a single feature space. However, the values are lower compared to the combination of exact divergences. Hence, we propose an alternative method in determining the candidate objects. In the proposed method, the Euclidean distances are not combined. Instead, we generate a set of candidate objects with respect to each feature space, see Figure 4.9.

$$A = \{a_1,\ a_2,\ \ldots\ ,\ a_m\}$$
$$B = \{b_1,\ b_2,\ \ldots\ ,\ b_m\}$$
$$C = \{c_1,\ c_2,\ \ldots\ ,\ c_m\}$$

$$candidates = \{A \cap B \cap C\}$$

where A, B and C are the candidate sets from each feature space, m is the filter size and the final candidate set is formed from the intersection of A, B and C. The size of the final candidate set can vary from a minimum of m (if the objects in a set are repeated in other sets) to a maximum of $3m$ (if no object is repeated for other candidate sets). This procedure, which we will call as *dynamic* filtering, exploits the computation time savings in the filter step while recognizing the advantage of each of feature space.

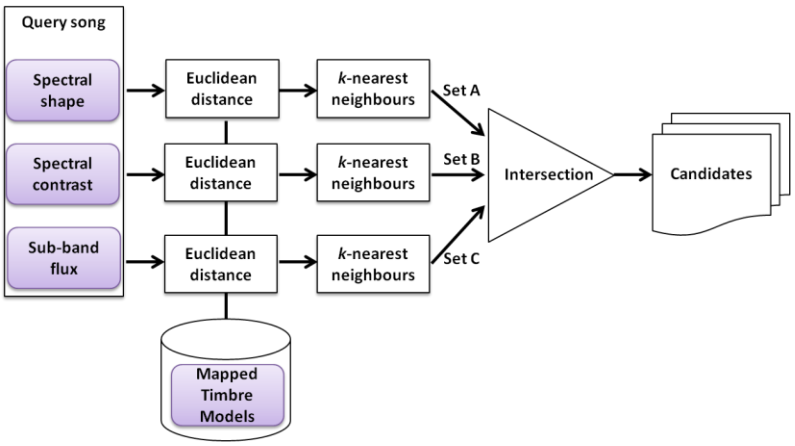
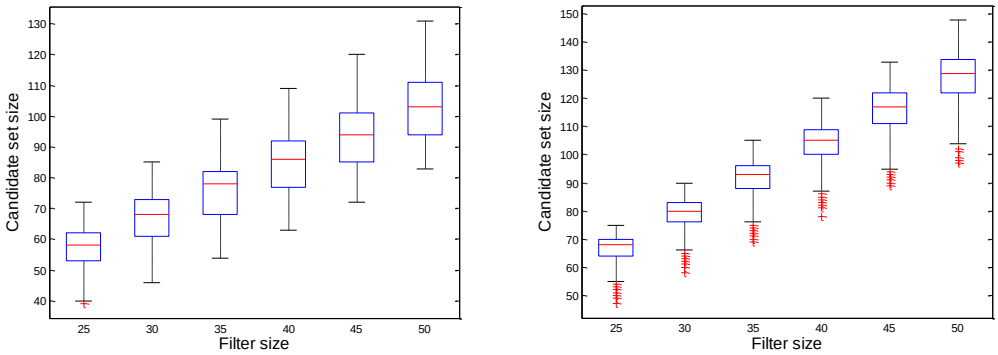


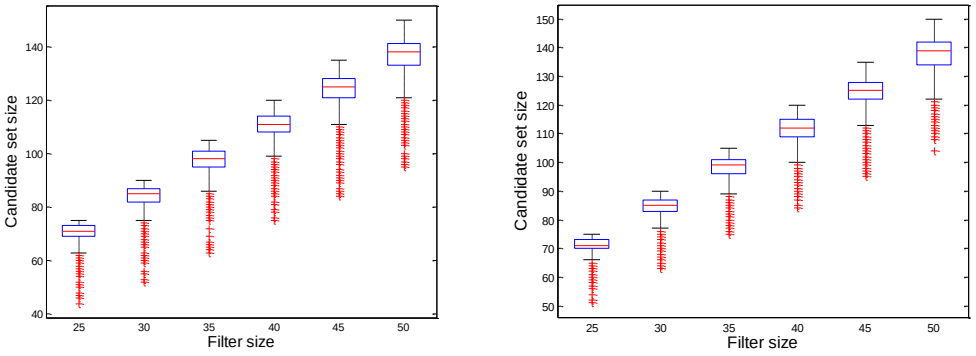
FIGURE 4.9: Block diagram of the filter step using dynamic filtering.

Figure 4.10 shows the box plots of the actual candidate sizes for all the collections. On each box, the median is indicated by the red line, the edges of the box are the 25th and 75th percentiles, the whiskers extend to the most extreme data points not considered outliers, and outliers are plotted individually[35]. The wide range in the plots implicate that each feature space returns a different set of candidates. In fact, the median values are often more than twice the value of the filter size.

[35] http://www.mathworks.co.uk/help/stats/boxplot.html



(a) DB-S (b) GTZAN



(c) 1517-Artists (d) Unique

FIGURE 4.10: Box plots of the range of candidate size for varying filter sizes.

4.3.2.2 Refine Step

After the candidate songs are determined, the computation of the exact divergence is the same as described in Section 3.4.2. The block diagram of the refine step is shown in Figure 4.11. The only difference of the refine step with full scan is that the similarity computations are performed on a fraction of the size of the dataset. Thus, the response time is significantly decreased.

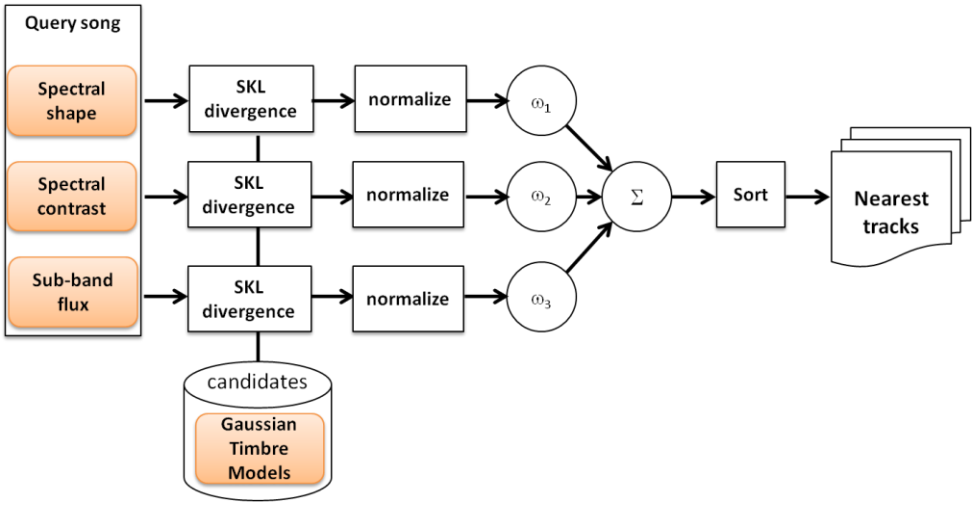


FIGURE 4.11: Block diagram of the refine step.

4.4 Experiments

The objectives of the experiments performed in this section are the following:

1. to determine how the retrieval performance varies for different filter sizes and datasets
2. to measure the response times for different configurations and datasets

The choice of performance measurements should reflect the requirements of a person doing a search. As mentioned before, the user is likely most interested on the first few items of the results, e.g. twenty songs. They may look deeper into the results, but are not likely to go beyond twenty songs. Hence, we will use the precision at 20 items (P_{20}) as an evaluation measure.

Evaluation of response times was discussed in Section 3.3.3.4. The measure used was a wall clock time over one hundred searches. Response times for a linear sequential scan are given as a baseline. A response time below ten seconds is considered an acceptable response time.

Results are shown for the full scan baseline system (MFCC), full scan using combined features, FastMap with static filtering, and FastMap with dynamic filtering.

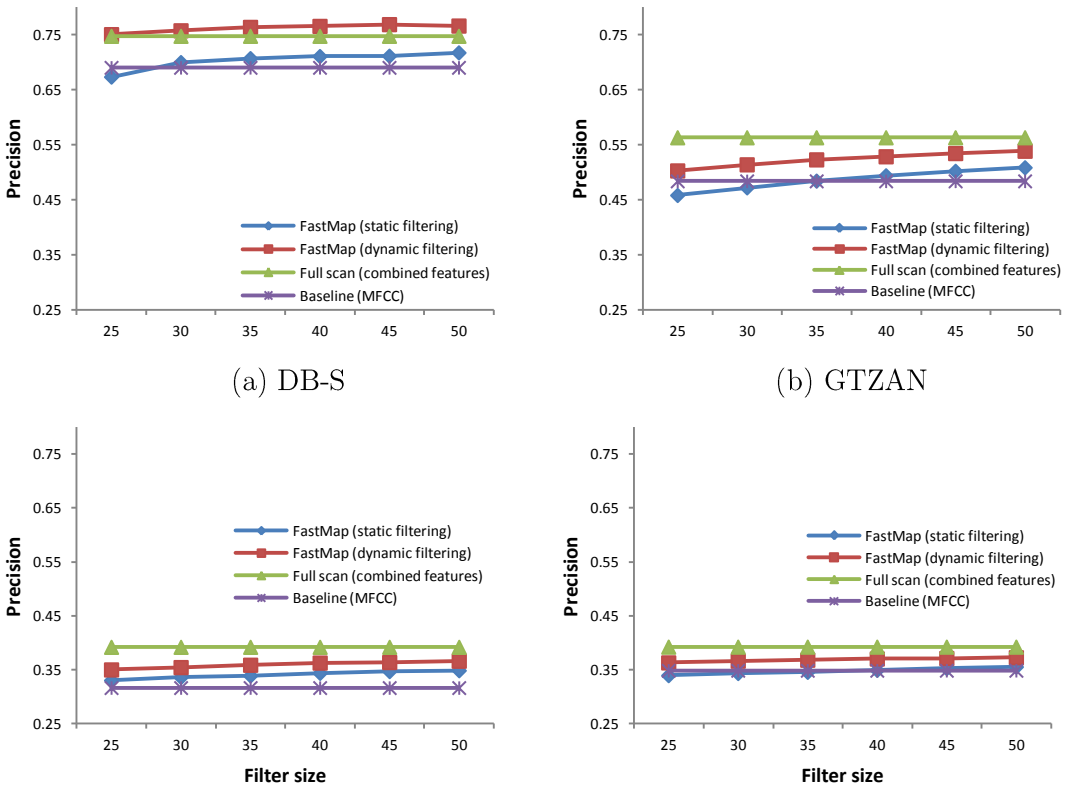
4.4.1 Retrieval performance

The precision at 20 items for different datasets are plotted in Figure 4.12. The filter size is varied from 25 to 50 songs. The precisions of the full linear scan system (green triangle) and baseline system (purple asterisk) are plotted to serve as the upper and lower bound of the results. We can see that FastMap with dynamic filtering (red square) consistently outperforms FastMap with static filtering (blue diamond) for a given filter size. The average difference in their precision is 0.02 using 1517-Artists and Unique datasets, 0.04 using GTZAN dataset, and 0.06 using DB-S dataset.

As expected, the precisions of the systems that used FastMap increases as the filter size increases. FastMap with dynamic filtering consistently outperforms the baseline system for all filter sizes. Meanwhile, the precisions of FastMap with static filtering are sometimes lower than the baseline system in the lower range of filter size. In both cases, results clearly validate the potential of the filter-and-refine approach despite the fact that FastMap was not able to preserve the original proximities between songs. This is further highlighted in the results using the DB-S dataset wherein FastMap with dynamic filtering even surpasses the performance of the full scan system using combined features. There are two possible reasons for this. First, the filter step effectively eliminates non-relevant songs by forming the candidate set as an aggregate of similar songs from each feature space. Second, the data space of the candidate set is different from full scan. Hence, the normalization procedure uses parameters from the smaller, more relevant candidate set which may help improve the results.

Now, let's take a look at the values of the precisions for each dataset. Consider the values for the full scan system using combined features. The highest value is observed for DB-S dataset (0.75), followed by GTZAN (0.56), while 1517-Artists and Unique come last (0.39). The difference in the values is attributed to the degree of homogeneity of the songs within a dataset. The DB-S dataset contains similar sounding songs within a given genre, particularly for pop rock and hard rock genres where there are 10 songs from every artist, and uniform, distribution of songs among genres. In contrast, 1517-Artists and Unique datasets have diverse artists within each genre and uneven distribution of songs

among genres. The author also did not verify if each song in these two datasets were appropriately tagged. Assuming that our system returns a set of timbrally similar songs to a query, it is possible that some of the songs belong to a different genre to the query. Hence, the construction of a dataset may affect performance values.



(a) DB-S

(b) GTZAN

(c) 1517-Artists

(d) Unique

FIGURE 4.12: Precision at 20 using filter-and-refine implementations for varying filter sizes.

4.4.2 Runtime

Measurements for runtimes are not as precise and repeatable as precision measurements. There are many variables including the hardware, operating system, development software, dataset and queries. Hence, we are more concerned with the general trends rather than accept the runtimes as absolute results.

Experimental set-up

The experiments were performed on a standard desktop computer with an Intel Xeon W3520 2.67 GHz processor and 12GB of RAM. The system has one Western Digital WD5000AKS SATA drive with the following specifications: 126 MB/s data transfer rate and 8.9 ms average seek time. The operating system was Windows 7 64-bit and the programs were developed using Matlab 7 (R2009b).

As discussed in Section 2.3.3, each feature is modelled as a single multivariate Gaussian distribution with full covariance matrix. Thus we only need to store the mean vectors and the covariance matrices from the feature vectors of each song. To expedite the computation of the Kullback-Leibler divergences, we also compute and store the inverse of the covariance matrices. When the features are embedded to the Euclidean space, the mapped vectors are stored as well. All these components occupy around 20 kB per song. For comparison, the average size of a 30-second wav file is 1.26 MB. The largest dataset used for testing contains 2847 songs. The corresponding features occupy almost 60 MB of space.

The runtime of the full scan system includes the processing time of computing the distance matrices from the feature spaces using SKL divergence, normalization of the distance matrices, and summing the normalized distance matrices. The size of the datasets for full scan system is fixed. Hence, the runtimes can be computed once for each dataset.

The runtime of the filter-and-refine system includes the processing time for determining the candidate songs using the Euclidean distance, computing the distance matrices from the feature spaces using SKL divergence, normalization of the distance matrices, and summing the normalized distance matrices. For static filtering, the size of the candidate set is fixed. Hence, the runtimes can e computed once for each dataset. For dynamic filtering, the size of the candidate set varies. Thus, a set of 100 queries was used for runtime testing. The results are then averaged. The queries were also shuffled so that

songs from the same genre did not appear consecutively. This minimizes the likelihood of a disk page being stored in the cache.

Full scan

In our initial measurements using the full scan system, we determined that the system is very slow. Even with a very small dataset of 250 songs, the runtime is more than 30 seconds. When the size of the dataset increases to 1000, the runtime becomes more than 600 seconds. Hence, it is impractical to apply this approach to a commercial system that contains a much larger dataset. To identify the bottleneck, we used the Matlab *profile* function to characterize the execution time of the algorithm.

Figure 4.13a shows the total execution times of our timbre similarity function for a dataset with 250 songs, including the parent and child functions, the corresponding number of calls, and the execution times. The second row lists the parent function called to compute the exact divergence between songs and to normalize the resulting distance matrix. The child functions are listed below the parent function. The child functions are other functions that are called within the parent function. For example, the figure on the left is the profile of the function before optimization. The execution time (encircled in red) of the whole algorithm is 43 seconds. The second row of the profile shows that the child function *squeeze* takes 22 seconds to execute, more than half of the whole execution time. This is the bottleneck of the algorithm.

The *squeeze* function is essential as it is used to remove the singleton dimension of a matrix, see Figure 4.14a. The covariance and inverse covariance matrices for each song are stored in the disk. If there are N songs and D-dimensional features, then the features are stored as a 3-d array $M(N{:}D{:}D)$. For a song index i, we can access the feature matrix $M(i{:}D{:}D)$. The *squeeze* function removes the singleton dimension to produce the feature matrix $I(D{:}D)$.

Function Name	Calls	Total Time
ComputeKLDistance_MFCC_spec_SC_Sbf_orig	1	43.208 s
squeeze	747000	22.104 s
trace	280125	3.474 s
Norm_dist_mat_1st_row	3	0.098 s
std	750	0.057 s
var	750	0.053 s
mean	750	0.022 s
datestr	2	0.021 s
num2str	1	0.021 s
timefun\private\dateformverify	2	0.018 s
timefun\private\formatdate	2	0.018 s
...Distance_MFCC_spec_SC_Sbf_orig>mydisp	7	0.011 s

(a) before optimization

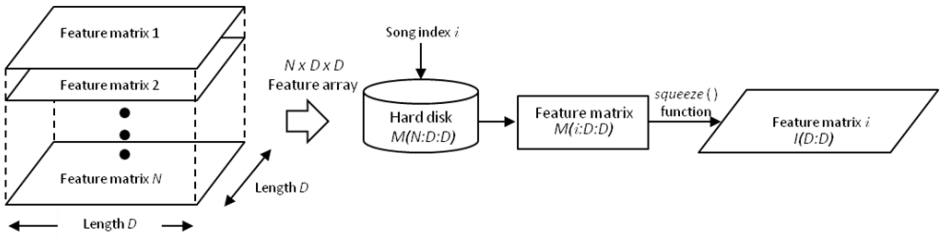
Function Name	Calls	Total Time
ComputeKLDistance_MFCC_spec_SC_Sbf	1	15.874 s
trace	280125	3.384 s
Norm_dist_mat_1st_row	3	0.097 s
std	750	0.065 s
var	750	0.054 s
datestr	2	0.021 s
mean	750	0.019 s
timefun\private\dateformverify	2	0.018 s
timefun\private\formatdate	2	0.017 s
...uteKLDistance_MFCC_spec_SC_Sbf>mydisp	7	0.010 s
now	2	0.003 s
datenum	2	0.002 s

(b) after optimization

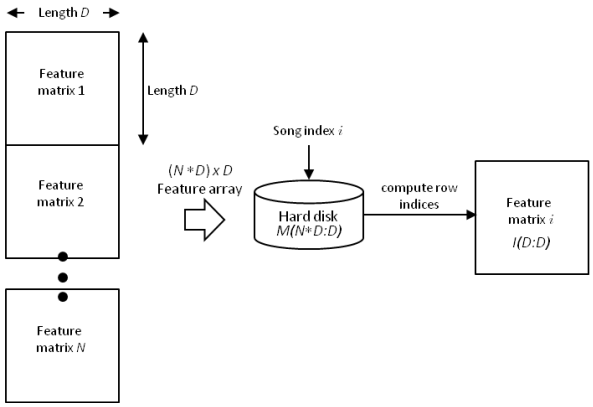
FIGURE 4.13: Execution time of the timbre similarity function before (left) and after optimization (right); evaluated using the DB-S dataset.

One solution to avoid using the *squeeze* function is to store the feature matrices as contiguous 2-*d* arrays $M(N*D{:}D)$, see Figure 4.14b. We can easily compute the corresponding row indices of the feature matrix given a song index i, e.g. $(i\text{-}1)*D{+}1{:}i*D$. With this optimization the execution time of the similarity function was reduced significantly, see Figure 4.13b. In this case, the execution time (encircled in green) is 16 seconds. For a dataset size of 250 songs, this means that it takes more than 60 ms to compute the timbre similarity between two songs using the Kullback-Leibler divergence.

The average runtimes for the different datasets using the full scan system are shown in Figure 4.15. The red line indicates the linear reference from the ratio 250:14, where 250 is the size of the smallest database with a corresponding average runtime of 14 seconds. It can be observed that the runtime increases faster than the linear reference. Moreover, the values are well above our target runtime of 10 seconds. Thus, the full linear scan approach is not suitable for commercial applications.



(a) Feature matrices stored as 3-dimensional arrays



(b) Feature matrices stored as contiguous 2-dimensional arrays

FIGURE 4.14: Block diagrams of storage and retrieval of feature matrices.

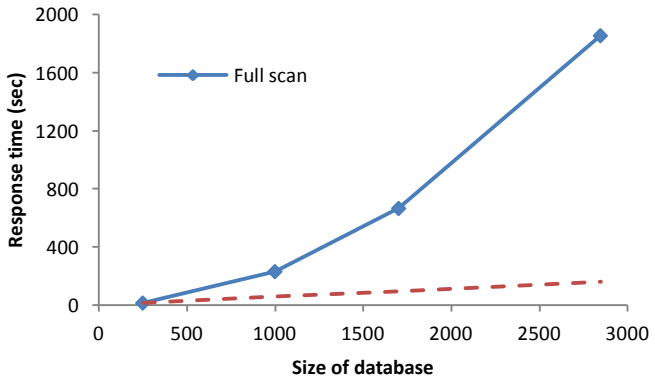


FIGURE 4.15: Runtime in seconds for full scan system.

FastMap

The implementation of a filter-and-refine system using FastMap is described in Section 4.3. It is based on using Euclidean distance to quickly return candidate songs for a given query then use the exact divergence to give more accurate results.

From the tabulated runtimes in Table 4.3, we can see that the filter-and-refine approach is much faster than performing a full scan. Based on the results, the runtimes of the filter-and-refine systems are below 5 seconds.

We also compare the runtime of static filtering against dynamic filtering when the filter size is varied, see Figure 4.16. As expected, the runtime increases as the filter size increases. However, the increase is more noticeable for dynamic filtering than static filtering. For a dataset of 250 songs and filter size of 30, the runtimes of static and dynamic filtering are 0.28 and 1.1 seconds respectively. When the size of the dataset is increased to 1000, the runtimes are now 0.42 and 1.7 seconds respectively.

What is the expected runtime of the filter-and-refine system if the size of the dataset increases to hundreds of thousands or even millions? We cannot give a definite answer given that we only have a limited dataset. However, we know that for a given filter size, the maximum size of the candidate set produced is fixed. Hence, the processing time for the computationally intensive refine step is also capped, regardless of the size of the dataset. On the other hand, the processing time for the filter step is dependent on the dataset size since it is at the front end. Nevertheless, the Euclidean distance can be computed relatively fast. Therefore, the filter-and-refine system has the potential to overcome the scalability problem.

Table 4.3: Runtime in seconds for full scan and filter-and-refine (Filter Size = 30, 40, 50) systems.

Dataset size	*Full scan*	*Static filtering*			*Dynamic filtering*		
		30	40	50	30	40	50
250	14.02	0.28	0.45	0.67	1.1	1.7	2.44
1000	230.89	0.42	0.58	0.79	1.7	2.73	4.02
1702	665.94	0.57	0.72	1	1.93	3.17	4.63
2847	1856.36	0.79	1.07	1.28	2.2	3.38	4.92

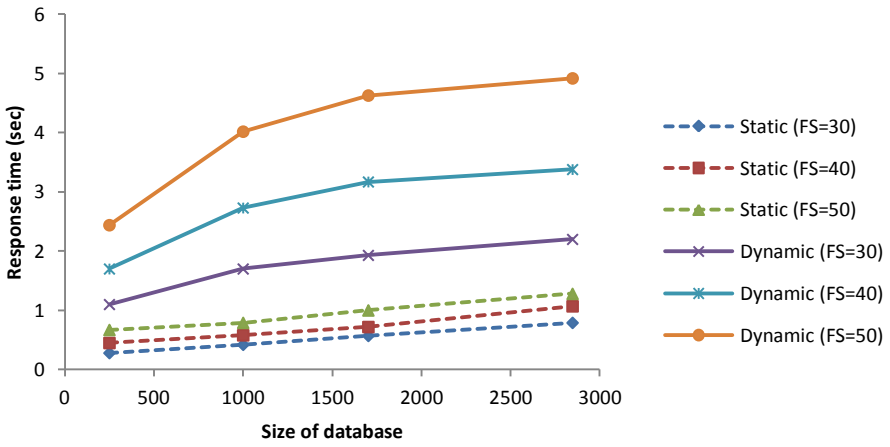


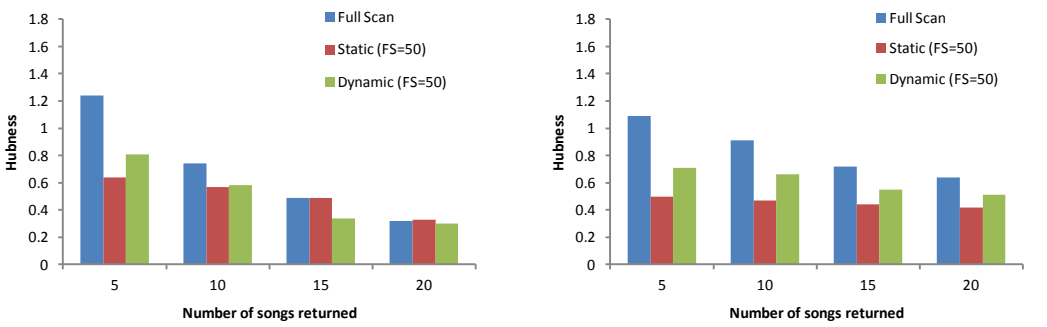
FIGURE 4.16: Runtime in seconds for filter-and-refine systems using different filter sizes (FS).

4.4.3 Hubs and orphans

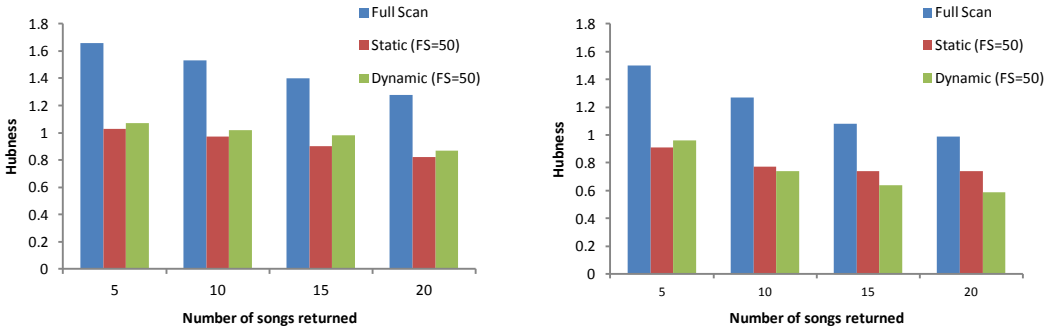
In Section 3.3.3.2, we discussed hubs and orphans which are machine learning problems that exist in high dimensional spaces. Hubs are always similar while orphans are never similar songs. We have shown that these can be reduced by normalizing the distances from each feature space before combining them. We want to compare the hubs and orphans produced by the filter-and-refine systems with the original full scan system.

Figure 4.17 shows the hubness values for the full scan system (blue bar) and FastMap system with static (red bar) and dynamic filtering (green bar) where the filter size is 50. In general, the filter-and-refine systems are able to further reduce the hubness. For the DB-S dataset, the improvements are only significant when the number of returned items is 5. For the other datasets, the hubness is reduced by at least 20%. The FastMap with static filtering produced the least amount of hubness using GTZAN and 1517-Artists datasets. However, there is no clear trend for the DB-S and Unique datasets.

In Figure 4.18, we compare the percentage of files within a dataset that are orphans. For all datasets, the FastMap with dynamic filtering produced the least number of orphan songs while the FastMap with static filtering produced the most number of orphan songs.

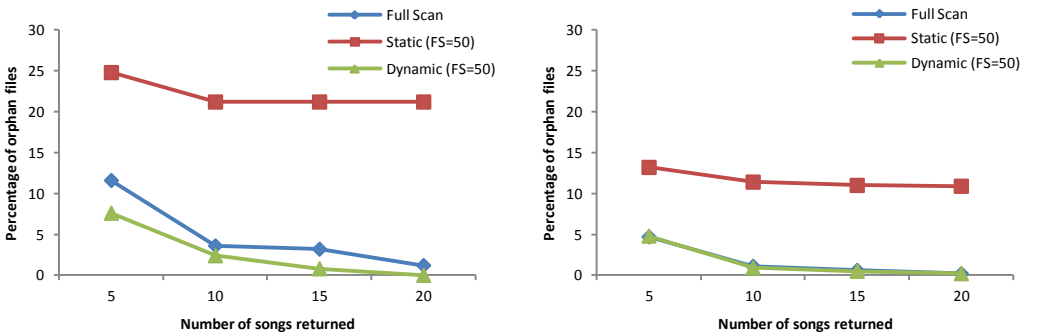


(a) DB-S (b) GTZAN

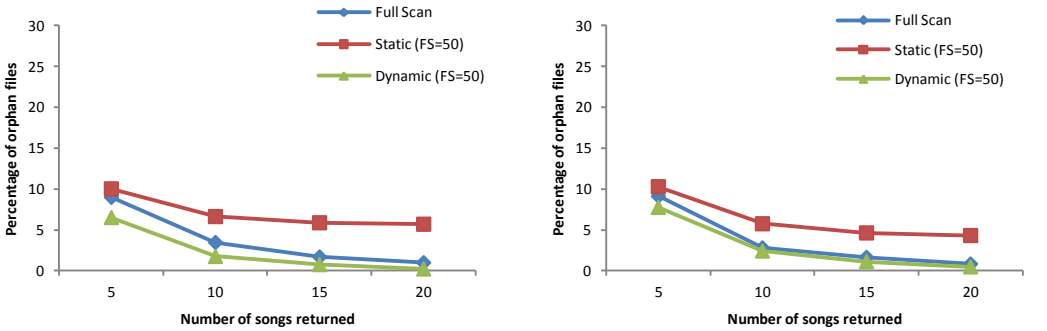


(c) 1517-Artists (d) Unique

FIGURE 4.17: Comparing hubness values of the filter-and-refine systems against the full scan system



(a) DB-S (b) GTZAN



(c) 1517-Artists (d) Unique

FIGURE 4.18: Comparing percentage of orphan files of the filter-and-refine systems against the full scan system

4.4.4 Summary

We adopted the filter-and-refine method to address the scalability problem in content-based audio retrieval. We modified the FastMap algorithm and integrated it to our system to embed the audio features in Euclidean space. Thus, the filter step involves computing the Euclidean distance between a query song and the songs in a dataset to quickly return a set of candidate songs. We proposed two variants of the filtering step: static and dynamic filtering. Static filtering is based on the metric properties of the Euclidean space to produce a fixed number of candidate songs. Dynamic filtering is based on exploiting the advantages of the feature spaces to return a variable number of candidate songs. Once the candidate sets are returned, the refine step involves computing the computationally intensive symmetrised KL divergence.

The full scan system using combined features and the two variants of the filter-and-refine systems were evaluated in terms of precision, runtime, and capability to reduce the presence of hubs and orphans. Based on the results, the full scan system produced the best precisions on three out of four datasets. However, it has the worst runtime making it unsuitable for large datasets. The filter-and-refine system with static filtering has the best runtime. It is also the best system that reduced hubs on two datasets. The filter-and-refine system with dynamic filtering produced the best precision for the DB-S dataset. Its runtime is not as fast as static filtering but is well below our target. It is also the best system in reducing orphans.

Overall, the filter-and-refine system with dynamic filtering is the best approach for an effective and efficient audio retrieval system using timbre similarity. The next step is to compare our system with the state-of-the-art by submitting to the MIREX audio music similarity task. We do not expect our submission to be the best performing system since it is developed to estimate timbre similarity only. However, the results can give us some indication on the correlation between timbre similarity and the concept of audio similarity. Aside from retrieval quality, the MIREX evaluation also measures other objective statistics to evaluate the submissions.

4.5 MIREX 2013 audio music similarity task results

This section presents the evaluation results of our submissions to the 2013 Music Information Retrieval Evaluation eXchange (MIREX). The annual evaluation is conducted by a team from the International Music Information Retrieval Systems Evaluation Laboratory (IMIRSEL) Project in the Graduate School of Library and Information Science at the University of Illinois. The MIREX results are important as it compares our algorithms with the state-of-the-art. It also assures that the different algorithms are compared fairly using a set of standardized setup and evaluation metrics.

We submitted two algorithms for the audio music similarity task. The first algorithm (DM1) is the full linear scan implementation of our system, see Figure 3.12. The output of this algorithm is a *dense* distance matrix where all pair-wise distances are included. The second algorithm (DM2) is the filter-and-refine implementation with dynamic filtering, see Figure 4.7 and Figure 4.9. We set the filter size to 50 songs. The output of this algorithm is a *sparse* distance matrix where only the distances of the top 100 results for each query in the dataset are returned.

Each submission was given 7000 songs chosen from IMIRSEL's "uspop", "uscrap", "american", "classical" and "sundry" datasets. Each system then returned a distance matrix. Fifty songs were randomly selected from the 10 genre groups (5 per genre) as queries and the first 5 most highly ranked songs out of the 7000 were extracted for each query (after filtering out the query itself, returned results from the same artist were also omitted). Then for each query, the returned results (candidates) from all participants were grouped and evaluated by human graders using the Evalutron 6000 grading system (Gruzd et al. 2007). Each individual query/candidate system was evaluated by a human grader with no knowledge of which submission returned the songs.

We briefly describe each of the music genres used in the evaluation, particularly on the instruments used. This will help us understand the results in the latter part of this section.

- Baroque, Classical, Romantic: These are art music limited to the high-culture tradition of Western Europe. Baroque music covers the period marked off the years 1580-1750 (Hill 2005) whereas Classical and Romantic music spans the years 1789-1914 (Blume 1972). They follow a Western staff notation that to prescribe to the performer the exact execution of a piece of music. It does not allow *improvisation* and *ad lib*, which can be heard in jazz and pop music. The symphony orchestra that plays the music includes members of the string, woodwind, brass, and percussion families. During the Baroque period, the main keyboard instruments used were harpsichord and pipe organ. In the classical period, the piano became the predominant keyboard instrument.

- Blues: The guitar is the main instrument used. It is played with a technique that is heavily dependent upon electronic amplification and manipulation. The accompanying band includes a prominent rhythm section of drums, electric bass, piano, and often electric piano and organ. There can also be supporting horns such as saxophones, trumpets, and trombones. The singing involves exaggeratedly stylized vocal delivery, with shouting and moaning (Kingman 1990).

- Country: The dominant instrument in country music is the fiddle. The music is characterized by straight, penetrating, vibrato-less tone and the sliding up into the longer held notes. The rest of the accompanying stringed instruments are plucked or strummed. These include the banjo, guitar, and mandolin. Later addition to the ensemble is the steel guitar, with its sliding, wailing sound. The singing style used in country music is clear, vibrato-less tone analogueous to that of the country fiddle (Kingman 1990).

- Jazz: Jazz is a *way* of playing and singing. The vital feature of jazz method is improvisation. The musicians are free to invent their own melodic lines that fit a given harmony and form. The "front line" instruments include saxophones, trumpets and trombones. They create the melody, or the simultaneous layering of melodies. The rhythm section includes piano and drums, and sometimes banjo. Their job is to keep the beat going and to outline the harmonies. The singing exhibits the same fluidity and virtuosity as heard in instrumental solos. It may also include "scat" singing, i.e. vocalizing nonsense syllables (Kingman 1990).

- Rock (rockroll): Like the blues, rock is a music for the singer (in contrast to the more instrumental jazz) backed up by an ensemble dominated by one or more guitars, a bass, and sometimes piano or other keyboard instrument, all driven by a strong relentless beat from the drummer. The common practices of adding a saxophone and a backup vocal are influences from rhythm-and-blues. It has developed many sub-styles. For example, "folk rock" will revert back to acoustic stringed instruments; "jazz rock" will introduce saxophones and brass, etc (Kingman 1990).

- Metal: It is a sub-genre of rock music. The essential feature in heavy metal is power, expressed as sheer volume. The lead instrument is the guitar, played in highly amplified distortion. Similar to blues-based guitarists, heavy metal guitarists are required to demonstrate technical proficiency. Hence, guitar solos are a common feature in this type of music. The guitar is moved along by a beat on a set of drums. The distinctive low frequency sound provided by the bass drum is enhanced by the electronic bass guitar. In heavy metal, the vocalist must also sound as powerful as the instruments (Weinstein 2000).

- Rap/hip-hop (raphiphop): Its principal characteristic is rapid-fire talking over a basic rock background. The style originated with black disk jockeys, *MCing* or *rapping* over the music (Kingman 1990). Standard musical instruments are not usually used. Instead, the disk jockeys use electronic devices – turntables, tape decks, synthesizers, mixers, drum machines, and samplers. (Crawford 2001)

- Electronic dance (edance): The main purpose of this music is for dancing. Hence, more emphasis is given on rhythm. The tracks are constructed using multi-track recording, signal processing, sequencing and sound synthesis. Multi-track recording allows the user to record each distinct instrument or voice separately. The individual tracks are fed to a multichannel mixer to balance the signal from each track. Signal processing modifies and enhances one or several tracks. Sequencing through hardware or software application handles the playback of instruments. Sound synthesis refers to generation of sound through electronic machines (Warner 2003).

For each query/candidate pair, graders provided two scores. Graders were asked to provide one *categorical broad score* {Not similar=0, Somewhat similar=1, Very similar=2}, and one *fine score* {in the range from 0 (failure) to 100 (perfection)}. Finally, the Friedman test is applied to the results to conduct a statistical pair-wise comparison of the submission results and to determine if any differences in ranking are significant.

Aside from the subjective tests, the following objective statistics are also derived from the distance matrix:

1. Average % of genre, artist, and album matches in the top 5, 10, 20 and 50 results (Precision at 5, 10, 20 and 50)
2. Average % of genre matches in the top 5, 10, 20 and 50 results after artist filtering of results
3. Percentage of songs never similar (never in the top 5, 10, 20 and 50 results)
4. Maximum number of times a song was in the top 5, 10, 20 ad 50 result list

5. Percentage of song triplets where triangular inequality holds

4.5.1 Current techniques

There were eight submissions for 2013 audio music similarity task. The participants and their respective submission codes are tabulated in Table 4.4. The following paragraphs describe the features and distance functions of the submitted algorithms:

- PS1: This submission is a variant of the algorithm described in (Pohle et al. 2009). The algorithm has two major components, a rhythm component and timbre component. The rhythm component is based on the cent-scale representation of Fluctuation Patterns (Pampalk 2006b). The *cent* is a logarithmic unit of measure used for musical intervals. For example, a musical octave is divided into 12 semitones of 100 cents each. The timbre component consists of MFCCs, spectral contrast and two other feature values estimating the amount of harmonic and percussive elements as discussed in (Ono et al. 2008). The rhythm component is summarized by taking the mean of the features whereas the timbre component is represented by a single Gaussian. The rhythm distance is computed by Euclidean distance whereas the timbre distance is estimated by symmetrised Kullback-Leibler divergence. The distances are normalised then combined linearly with equal weights.

- RA1: Unlike most submissions that are based on timbre and rhythm features, this submission is based on chroma feature extraction using a continuous wavelet transform and modelling music structures using hidden Markov models (Aliyev 2013). Chroma features are the projections of the audio spectrum onto 12 bins representing the 12 distinct semitones (chroma) of the musical octave. The time series of chroma features are extracted using continuous wavelet transform. Each feature is then quantized to the nearest element from the codebook. The codebook has been trained with 100 hours of music and clustered using K-means algorithm. The next stage is the estimation of the probabilities that the codeword indices are matched by one of the 347 music structure models from SALAMI project (Smith et al. 2011). Thus, each song is represented by a vector containing 347 probability values and the similarity between songs can be computed by Manhattan distance between their vectors.

- SSPK1: This submission used block-level features for genre classification, tag classification and music similarity estimation. The block features are based on the cent-scaled magnitude spectrum. This is obtained by mapping the Short Time Fourier Transform magnitude spectrum with linear frequency resolution onto logarithmic cent scale. The compressed magnitude spectrum is then transformed to a logarithmic scale. Finally, the spectrum is made intensity-invariant by removing the mean computed over a sliding window from each audio frame. Eight

block-level features representing timbre and rhythm are extracted for every song. Please refer to Seyerlehner et al. (2010) for more details on block-level features. In addition, they also extracted local single Gaussian model of timbre using a block of 100 MFCC frames. The last feature uses 200 Mel filters and 50 MFCCs to precisely model the spectral envelope of an audio frame. The features are then summarized using their mean and variance. To estimate pair-wise music similarity, the Manhattan distance for each block-level features are computed. The distances are then Gaussian normalised, weighted then summed.

The descriptions for GKC1, GKC2, and SS2 were not provided by their respective authors.

TABLE 4.4: Participants for the 2013 MIREX audio music similarity task

Contributors	*Team code*	*Research Institution*
Franz de Leon, Kirk Martinez	DM1	University of Southampton
Franz de Leon, Kirk Martinez	DM2	University of Southampton
Aggelos Gkiokas, et al.	GKC1	Institute for Language and Speech Processing
Aggelos Gkiokas, et al.	GKC2	Institute for Language and Speech Processing
Dominik Scnitzer, Tim Pohle	PS1	Johannes Kepler University
Roman Aliyev	RA1	Belarusian State University
Klaus Seyerlehner, Markus Schedl	SS2	Johannes Kepler University
Klaus Seyerlehner, et al.	SSPK1	Johannes Kepler University

4.5.2 Subjective results

Figure 4.19 compares the resulting broad and fine scores of the submitted systems. SS2 achieved the best performance in both scores. This is closely followed by SSPK1 and PS1. Our submissions, DM1 and DM2, are shown in red bars. Overall, DM2 ranks fourth while DM1 ranks sixth. For both broad and fine scores, DM2 performed better than DM1. This exceeded our expectations since DM1 performs the computationally intensive full linear scan. However, we have already observed in our local experiments using the DB-S database that the filter-and-refine approach can outperform the full scan system.

To give further details on the results, the average broad and fine scores for every test genre are enumerated on Table 4.5 and Table 4.6. Suppose we set the minimum satisfactory level at 50%, i.e. minimum broad score of 1 and minimum fine score of 50. This means that for the 5 candidate songs returned by each algorithm, at least half are

judged similar to the query. The values from our submissions that satisfy this condition are shown in bold. At this level, DM1 performs well with baroque, blues, jazz and romantic genres. Meanwhile, DM2 performs well on all genres except baroque, metal, raphiphop and rockroll genres. It is interesting to note that no submission can surpass the arbitrary satisfactory level across all genres. All algorithms had difficulty returning relevant songs to the query songs from Rockroll genre. This shows the difficulty of audio music similarity task.

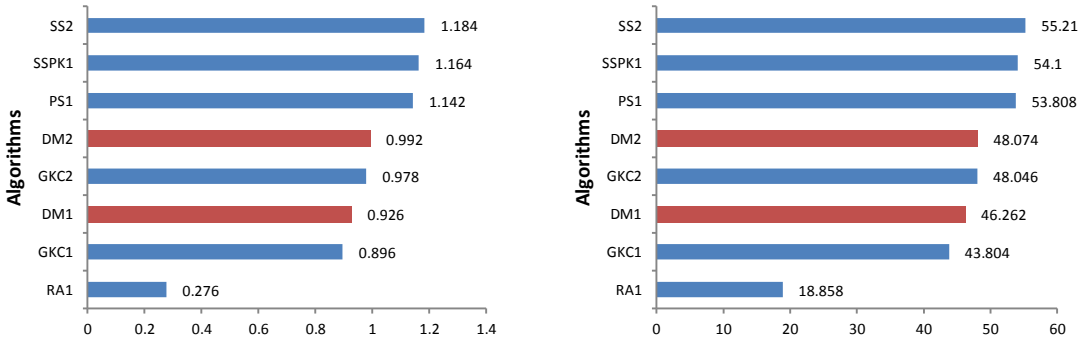
Table 4.5: Average broad scores for each genre

Genre	**DM1**	**DM2**	GKC1	GKC2	PS1	RA1	SS2	SSPK1
Baroque	**1.36**	**1.08**	1.12	1.12	1.42	0.04	1.42	1.52
Blues	**1.08**	**1.1**	0.86	0.92	1.24	0.68	1.24	1.26
Classical	**1**	**1.08**	0.9	0.9	0.96	0.04	1.1	0.96
Country	0.82	0.96	0.86	0.94	1.16	0.3	1.32	1.2
Edance	0.9	**1.1**	1.2	1.16	1.14	0.36	1.42	1.38
Jazz	**1.14**	**1.18**	1.24	1.4	1.36	0.38	1.4	1.46
Metal	0.7	**1**	0.58	0.76	1.14	0.46	0.92	1.02
Raphiphop	0.72	0.74	0.86	1.06	1.08	0.12	0.96	0.98
Rockroll	0.46	0.66	0.48	0.58	0.72	0.34	0.92	0.82
Romantic	**1.08**	**1.02**	0.86	0.94	1.2	0.04	1.14	1.04

Table 4.6: Average fine scores for each genre

Genre	**DM1**	**DM2**	GKC1	GKC2	PS1	RA1	SS2	SSPK1
Baroque	**57.22**	48.2	48.9	50.72	62.4	5.98	62.58	65.58
Blues	**55.54**	**54.66**	48.32	47.68	56.88	37.3	57.5	57.58
Classical	48.78	**51.8**	45.28	45.94	49.96	10.18	52.3	48.32
Country	46.72	**50.9**	46.72	52.16	56.48	24.14	63.4	59.88
Edance	48.14	**52.54**	53.6	53.7	55.02	24.68	62.44	61.34
Jazz	**60.96**	**62.2**	61.28	70.94	68.6	27.94	69.56	68.62
Metal	28.74	38.6	24.56	34.52	43.86	19.8	38.68	41.1
Raphiphop	39.3	42.12	45.48	52.34	52.88	15.94	50.66	50.04
Rockroll	25.06	29.26	23.02	28.38	33.52	15.72	40.68	38.16
Romantic	**52.16**	**50.46**	40.88	44.08	58.48	6.9	54.3	51.18

Applying Friedman's test on the fine scores indicates that the difference between the top 3 submissions {SS2, SSPK1, PS1} are not statistically significant[36]. In addition, Friedman's test on the broad scores reveals that the difference between the top 3 submissions and our submission (DM2) are not statistically significant. The advantage of the top 3 submission may be due to the fact that their systems include the rhythm component in feature extraction.



(a) Average broad score

(b) Average fine score

FIGURE 4.19: Results of the AMS systems based on the subjective test results.

4.5.3 Objective results

Ideally, it is desired that the full evaluation is performed by human test subjects. Unfortunately, this approach is impractical as it would require a large number of man-hours for the results to be reliable. For example, given that there are 50 query songs, 5 candidate songs per query, 8 algorithms, and 3 graders per query, then the maximum number of query result pairs that must be evaluated is 6,000, i.e. assuming there are no overlaps between the results from the different submissions. Suppose it takes roughly a minute to make a similarity judgment, it would require two hours of evaluation work from each of the 50 volunteer graders.

[36] http://www.music-ir.org/mirex/wiki/2013:Audio_Music_Similarity_and_Retrieval_Results

Neighbourhood clustering according to indexed metadata

Given the high man-hours required and the relatively small scope provided by the subjective tests, evaluations based on statistics of the number of candidates having the same metadata (genre, artist, or album) amongst the k most similar candidates for each query are also presented. Figure 4.20 and Figure 4.21 plots the neighbourhood clustering statistics with respect to artist and album matches. These statistics are strong indicators of timbre similarity based on the assumption that the results from the same artist or album are most likely timbrally relevant. Interestingly, the precisions of our submissions are comparable with the best performing systems. In fact, our DM2 submission has the highest precisions for both artist and album neighbourhood clustering for the top 5 candidate songs. This implies that our filter-and-refine system has the highest probability of returning timbrally relevant songs on the top items of a returned list.

Figure 4.22 shows the average percentage of genre matches before artist filtering the results. At top 5 candidates, the performances of our submissions are as good as the best submission. The average precisions of DM1 and DM2 are 0.74 and 0.76 respectively while the best submission (SS2) has an average precision of 0.77. As the number of returned songs increases, the performance gap between the top three systems and our submissions increases. Moreover, it shows that the full linear scan system performs better than the filter-and-refine system. Figure 4.23 shows the average percentage of genre matches after artist filtering the results. In general, the precision values decreased significantly compared to the results before artist filtering. For example, the precision at top 5 results of DM2 decreased from 0.76 to 0.54. The top 3 systems {SS2, SSPK1, PS2} have similar precision values. The figure also shows that the performances of our submissions are similar, but the values are lower than the top 3 systems.

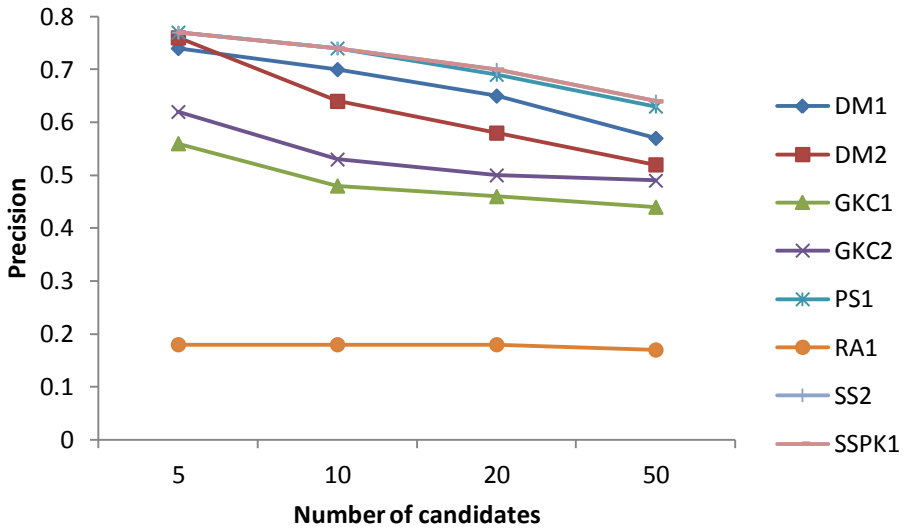


FIGURE 4.20: Average percentage of artist matches in the top 5, 10, 20 and 50 results.

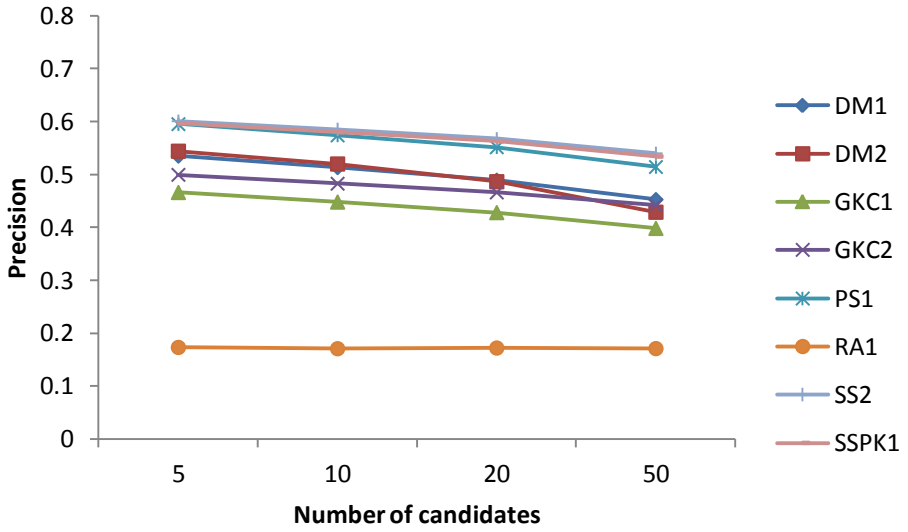


FIGURE 4.21: Average percentage of album matches in the top 5, 10, 20 and 50 results

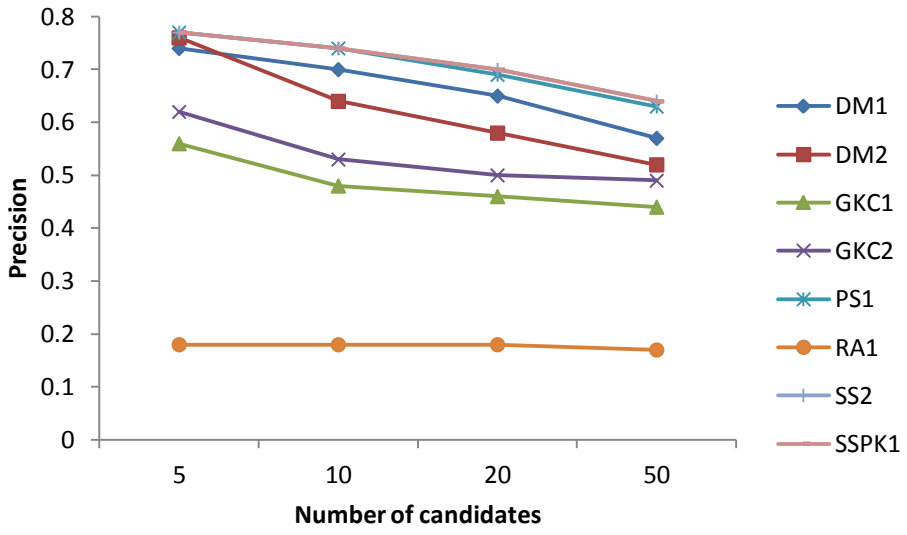


FIGURE 4.22: Average percentage of genre matches in the top 5, 10, 20 and 50 results before artist filtering

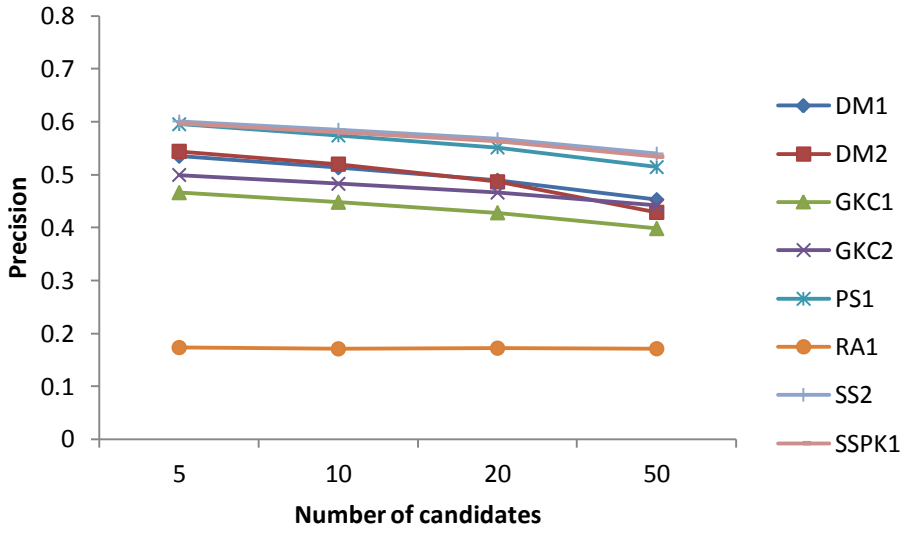


FIGURE 4.23: Average percentage of genre matches in the top 5, 10, 20 and 50 results after artist filtering

The average precision per genre in the top 5 results after artist filtering is shown in Figure 4.24. Again, suppose we set the minimum satisfactory precision at 0.5. At this level, our systems achieve a good performance in all genres except rockroll, edance, classical, and romantic. These results are inconsistent with the subjective results earlier. To resolve the discrepancy between the subjective and objective results, the artist filtered

genre neighbourhood confusion matrix at top 5 results for DM2 is tabulated in Table 4.7. The columns represent the genre of the query songs and the rows denote the genres of the candidate songs. For example, if the query song is metal, on the average 66% of the candidates are metal, 20% are rockroll, 4% are country, etc.

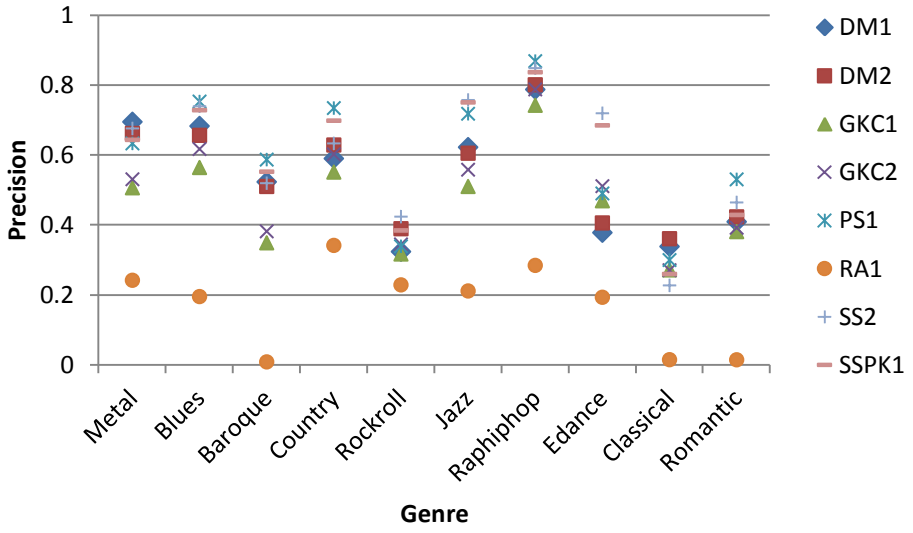


FIGURE 4.24: Average precision per genre at top 5 results after artist filtering of results

TABLE 4.7: Artist filtered genre neighbourhood confusion matrix at top 5 results (DM2)

query / response	Metal	Blues	Baroque	Country	Rockroll	Jazz	Raphiphop	Edance	Classical	Romantic
Metal	0.662	0.003	0.003	0.024	0.218	0.008	0.029	0.141	0.001	0.001
Blues	0.004	0.656	0.012	0.06	0.034	0.169	0.003	0.008	0.01	0.015
Baroque	0.002	0.027	0.51	0.01	0.003	0.014	0.001	0.007	0.208	0.22
Country	0.039	0.083	0.018	0.629	0.237	0.108	0.035	0.059	0.007	0.006
Rockroll	0.208	0.028	0.012	0.147	0.389	0.033	0.033	0.096	0.001	0.004
Jazz	0.012	0.157	0.02	0.085	0.041	0.605	0.012	0.044	0.015	0.009
Raphiphop	0.027	0.004	0.001	0.02	0.041	0.015	0.801	0.227	0	0.002
Edance	0.041	0.003	0.004	0.017	0.032	0.02	0.085	0.406	0.001	0.002
Classical	0.002	0.014	0.221	0.004	0.001	0.014	0	0.004	0.361	0.319
Romantic	0.003	0.024	0.199	0.005	0.004	0.014	0.002	0.007	0.398	0.423

We can observe some patterns from the confusion matrix. Metal is confused with rockroll and, country; blues is confused with jazz; baroque is confused with classical and romantic; edance is confused with raphiphop. Referring back to the genre descriptions given earlier, the results should be unsurprising knowing that some of the genres have the same style and instrumentation. In fact, some music genres naturally overlap. For example, the following comment from (Blume 1972) emphasizes this point regarding classical and romantic music:

> "There is neither a "Classic" nor a "Romantic" style in music. Both aspects and both trends are continually merging into one. And as there can neither be a clearly definable borderline between Classism and Romanticism nor a distinct chronology of when the one or the other begins and ends."

We can therefore cluster the related genres and tabulate the modified confusion matrix as shown in Table 4.8. Based on this table, the average genre clustering accuracy is 85%. Hence, effectiveness of our submitted systems in returning songs with similar timbre to a query is validated. The system works best with metal, baroque, classical, and romantic music where the clustering accuracy is more than 90%. However, it performed worst with edance music where the clustering accuracy is 63%. A possible reason for the low result is that timbre is not enough in retrieving edance music. In fact, we already observed this in the timbre space from our local experiment; see Figure 3.7, where electronica songs occupy almost the same space as the pop rock and hard rock songs. This observation is validated in Table 4.8 where edance is confused with metal, country, and rockroll cluster. Since edance is particularly made for dancing, a feature space that describes rhythm might help differentiate it better from other genres.

TABLE 4.8: Artist filtered, clustered genre neighbourhood confusion matrix at top 5 results (DM2)

query / response	Metal	Blues	Baroque	Country	Rockroll	Jazz	Raphiphop	Edance	Classical	Romantic
Metal, Country, Rockroll	0.909	0.114	0.033	0.799	0.844	0.149	0.096	0.296	0.009	0.011
Blues, Jazz	0.016	0.813	0.031	0.145	0.075	0.774	0.015	0.052	0.024	0.023
Baroque, Classical, Romantic	0.007	0.065	0.929	0.019	0.008	0.041	0.003	0.019	0.966	0.962
Raphiphop, Edance	0.069	0.007	0.006	0.037	0.073	0.035	0.885	0.633	0.001	0.003

Hubs and orphans

Aside from achieving high precisions, an effective retrieval system must also minimize hubs and orphans. The performance of the submitted systems in terms of the number of hubs and orphans produced are plotted in Figure 4.25 and Figure 4.26 respectively. Figure 4.25 shows that our submission DM2 consistently produced the least number of hubs whereas DM1 produces almost twice that of DM2. This can be attributed to the architecture of the filter-and-refine system wherein each feature returns an independent set of candidates. As we have observed in our local experiments, each of these features has a particular advantage towards certain genres. The net result is that there is a higher probability of returning relevant songs. This set-up combined with Gaussian normalization has the net effect of reducing the probability of hubs.

In terms of the percentage of orphans produced, Figure 4.26 shows that our systems have also minimized the probability of orphans. This time, DM2 has no clear advantage over DM1. Among the top performing systems in terms of precision, SS2 and SSPK1 are also able to reduce the hubs and orphans while PS1 performed worse than our submissions.

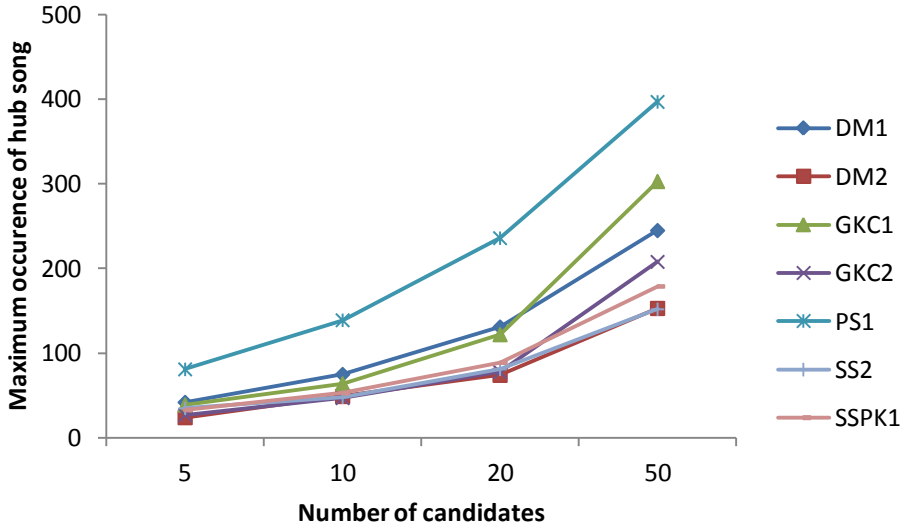


FIGURE 4.25: Number of hub songs in the top 5, 10, 20 and 50 results

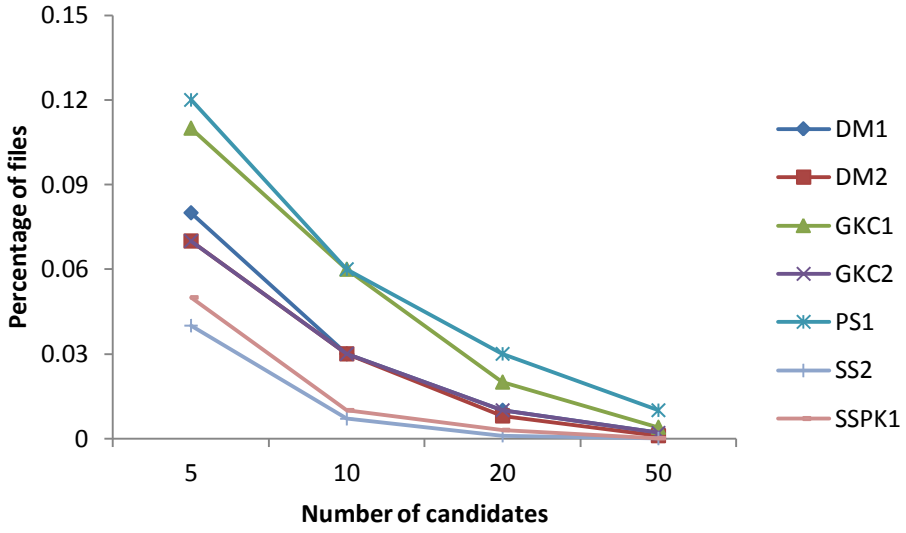


FIGURE 4.26: Percentage of orphan songs in the top 5, 10, 20 and 50 results

4.5.4 Summary

We submitted two systems to the MIREX 2013 audio music similarity and retrieval task: 1) full scan system using combined features (DM1), and 2) filter-and-refine system with dynamic filtering (DM2). The complete results for the two systems can be found on Appendix 4 and 5. The subjective results show that the performance of the filter-and-refine system is slightly better than the full scan system. The average broad and fine scores of our systems indicate that the degree of perceived similarity between a query

song and the top 5 returned songs is almost 50%. In comparison, the average scores for the best submission are almost 60%. However, the subjective test results do not tell the whole picture since they are dependent on the set of query songs used.

Objective statistics are derived from the distance matrices produced by the submitted systems. Based on the neighbourhood clustering statistics with respect to artist and album matches, our submissions are comparable with the best performing systems. In fact, DM2 has the highest neighbourhood clustering accuracy at 5 items according to artist and album. These statistics are strong indicators of timbre similarity based on the assumption that the results from the same artist or album are most likely relevant. This implies that our filter-and-refine system has the highest probability of returning timbrally similar songs at the top of a returned list.

The neighbourhood clustering statistics with respect to genre matches exposed our systems' strengths and weaknesses. The neighbourhood clustering accuracy at 5 items according to genre of DM1 and DM2 is 54%, while the best submission is 60%. From the genre neighbourhood confusion matrix, we observed that songs are often confused with other timbrally similar genres. If we cluster the related genres, the resulting genre neighbourhood clustering accuracy increases to 85%. Our system works particularly well with baroque, classical, and romantic music since they have a distinct orchestration. However, our system had difficulty with edance music. This agrees with our observations in our local experiments where edance is often confused with rock. Perhaps the addition of a feature space that describes rhythm can help minimize the confusion.

Overall, the results of the MIREX 2013 audio music similarity and retrieval task validated the effectiveness of our systems. Moreover, the architecture of our systems minimizes the probability of hubs and orphans. The system presented here is limited to estimating polyphonic timbre similarity using low-level features. The next challenge is to capture a more meaningful concept of "music similarity". This would require a deeper understanding of the processes in music cognition. It would be interesting to see how computational models that are capable of machine learning, such as artificial neural networks and *deep learning*, can be applied to music similarity tasks.

Chapter 5: Automatic Classification of Audio

Music classification is an important task in music information retrieval. Various classification problems have emerged during the last 10 years. The audio classification problem ranges from a more general genre classification problem to more specific tasks such as mood detection. In this chapter, the audio features and techniques used in Chapter 4 are applied to music genre classification.

The chapter begins by giving an overview of the different audio classification techniques. We select three methods commonly used in MIR literature: 1) k-nearest neighbour, 2) support vector machine, and 3) Gaussian mixture models. We discuss each method's advantages and disadvantages. The three methods are evaluated locally in terms of genre classification accuracy.

Finally, we present the results of our submissions to the MIREX 2013 audio classification task.

5.1 Classification algorithms and methods

A particular aspect of music classification is genre classification. The objective is to correctly categorize an unknown recording of a song with a music genre. Labels can be hierarchically organized in the dataset as genres and subgenres. Labelling can be used to enhance a musical document with high-level metadata or to organize a music dataset. At present, genre classification is still biased towards Western music. Thus, genre labels are the ones commonly used in Western music stores.

A number of different genre classification algorithms have been applied to low-level features calculated from music. Table 5.1 enumerates the most common approaches for music genre classification. The study by Tzanetakis and Cook (2002) was among the first to introduce the problem of music classification. The dataset used covered just a few classes and has some bias toward classical music and jazz. The lack of a standard set of genre is a typical problem of music classification, because the relevance of the different

categories is subjective, as well as the categories themselves. Hence, humans may classify a specific music to more than one genre.

Table 5.1: Genre classification methods

Techniques
Bayesian Model (Barutcuoglu et al. 2007)
Decision Tree (Anglade, Ramirez, and Dixon 2009)
Hidden Markov Model (Reed and Lee 2006)
Statistical Pattern Recognition (Tzanetakis and Cook 2002)
Wavelet Transformation (Grimaldi, Cunningham, and Kokaram 2003) (T. Li, Ogihara, and Li 2003)
Support Vector Machines (M. Li and Sleep 2005) (A. Meng and Shawe-Taylor 2005)
Taxonomy (T. Li and Ogihara 2005)
Multilabelling Classification (Lukashevich et al. 2009) (Wang et al. 2009)
Neural Networks (a. Meng, Ahrendt, and Larsen 2005) (Mckay and Fujinaga 2004)

The features used for genre classification are usually the same as the ones used in music similarity estimation. Most studies use content descriptors related to timbre. This choice is based on the fact that techniques try to classify short excerpts of an audio recording. In a study by Gjerdingen and Perrott (2008), it was found that humans can perform genre classification in as short as 250 ms. It was argued that timbre encompasses all the spectral and rapid time-domain variability in the acoustic signal. Such information can be highly indicative of particular genres. Other features, such as melody or rhythm cannot be derived from such short audio clips. In contrast, automatic classifiers work on longer samples that enable them to derive features other than timbre. The next sections discuss the approaches to genre classifications that are evaluated in this work.

5.1.1 *k*-Nearest Neighbours

The *k*-Nearest Neighbours algorithm (*k*-NN) belongs to a class called *lazy* classifiers. Lazy classifiers do not attempt to form a model of data and how it is divided into classes prior to classification. Instead, the classification procedure is executed at runtime. This is

a widely used classifier due to its simple implementation. On the other hand, it requires that large amounts of data be stored and a large number of computations be performed.

Figure 5.1 shows the block diagram of a genre classifier using k-NN. The first step in the implementation of the k-NN classifier is the construction of a dictionary composed of song-level features. The song-level features are the same single Gaussian models used in timbre similarity estimation. These features are labelled according to the genre they best represent. Thus, each genre has a set of single Gaussian distributions. The classification is done by comparing the song-level features from an untagged song with the features in the dictionary. Finally, classification is given depending on the genre that appears the most number of times among the k nearest songs.

Two systems described in Section 3.4.2 and Section 4.3.2 are used for comparing song-level features. These are the full linear scan system and the filter-and-refine approach with dynamic filtering. The variable that is varied in the experiments is the size of the neighbourhood (k) used to classify each song.

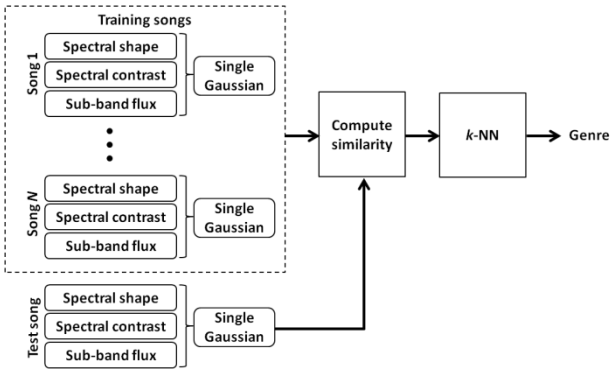


FIGURE 5.1: Block diagram of genre classification using song-level features and k-nearest neighbours.

5.1.2 Support Vector Machine

Support vector machine (SVM) (Boser et al. 1992) is a type of linear classifier that attempts to find the *hyperplane* that best separates observations or feature values pertaining to two different classes in multidimensional space. SVMs try to maximize the geometric margin between classes. Intuitively, maximizing the margin between classes

should lead to a robust classifier as small changes in data points should not lead to classification errors. The theory does not guarantee that the best hyperplane can always be found, but in practice, a heuristic solution can always be obtained.

Given a training set of instance-label pairs (x_i, y_i), $i=1,\ldots,l$ where $x_i \in R^n$ and $y \in \{1, -1\}^l$, the support vector machines require the solution of the following optimization problem:

$$\min_{w,b,\xi} \quad \frac{1}{2} w^T w + C \sum_{i=1}^{l} \xi_i \qquad 5.1$$

$$\text{subject to} \quad \begin{array}{l} y_i(w^T \phi(x_i) + b) \geq 1 - \xi_i, \\ \xi_i \geq 0 \end{array}$$

The hyperplane is described by $w^T \phi(x_i) + b = 0$ where:

- w is normal to the hyperplane
- $\frac{b}{\|w\|}$ is the perpendicular distance from the hyperplane to the origin

The training vectors x_i are mapped into a higher dimensional space by the function ϕ. SVM finds a linear separating hyperplane with the maximal margin in this higher dimensional space. $C>0$ is the penalty parameter of the error term. Furthermore, $K(x_i, x_j) \equiv \phi(x_i)^T \phi(x_j)$ is called the kernel function. This *kernel trick* extends the applicability of SVM to nonlinearly separable data. This means that by using the kernel function, only inner products of the mapped inputs in the feature space need to be determined without the need to explicitly calculate ϕ. There are four basic kernels:

1) linear: $K(x_i, x_j) = x_i^T x_j$

2) polynomial: $K(x_i, x_j) = (\gamma x_i^T x_j + r)^d, \gamma > 0$

3) radial basis function (RBF): $K(x_i, x_j) = \exp(-\gamma \|x_i - x_j\|^2), \gamma > 0$

4) sigmoid: $K(x_i, x_j) = \tanh(\gamma x_i^T x_j + r)$

where γ, r, and d are the kernel parameters.

Given the available kernel functions, the RBF kernel is preferred for several reasons. First, unlike the linear kernel, it can handle nonlinear relations between class labels and attributes. Second, the RBF kernel has fewer parameters than the polynomial kernel. Third, the RBF kernel has fewer numerical difficulties, i.e. $0<K_{ij}\leq 1$. For the polynomial kernel, K_{ij} approaches infinity or zero while the degree is large. Moreover, the sigmoid kernel is not valid under certain values of γ and r (Vapnik 1999).

Since genre classification is a multiclass problem and the SVM is a binary classifier, some modifications must be made. The most common approach is to reconfigure the single multiclass problem into multiple binary classification problems. In this work, the Directed Acyclic Graph SVM (DAGSVM) by Platt et al. 2000) is used. While other multiclass SVM approaches (e.g. one-against-all, one-against-one) may achieve better results, DAGSVM is more practical because it is faster to compute (Hsu and Lin 2002).

For an N-class problem, the DAGSVM contains $N(N\text{-}1)/2$ classifiers, one for each pair of classes. DAGSVM operates in a kernel-induced feature space and uses two-class maximal margin hyperplanes at each decision node of the Decision Directed Acyclic Graph (DDAG). The DDAG is obtained by training each *ij*-node only on the subset of training points labelled by i or j. Figure 5.2 shows the DDAG for four classes. A test point is evaluated against the decision node that corresponds to the first and last elements of the list. If the node prefers one of the two classes, the other class is eliminated from the list, and the DDAG proceeds to test the first and last elements of the new list. The DDAG terminates when only one class remains on the list.

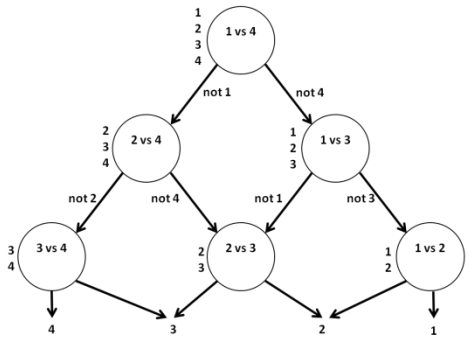


FIGURE 5.2: Decision Directed Acyclic Graph (DAG) for four classes (Platt, Cristianini, and Shawe-taylor 2000)

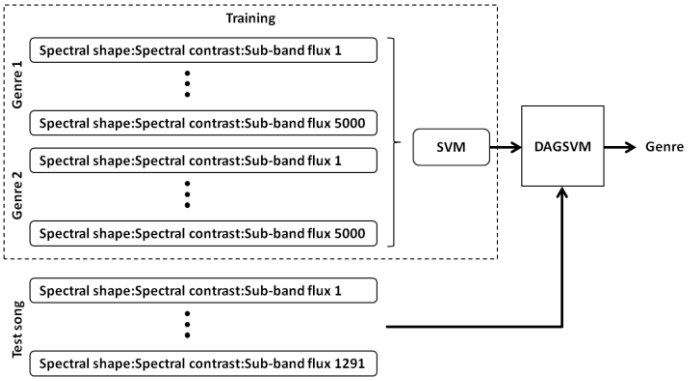


FIGURE 5.3: Block diagram of genre classification using DAG SVM.

Figure 5.3 shows the block diagram of the classifier using DAGSVM. Genre classification is performed at the frame level. We implement DAGSVM in MATLAB® using the SVM library developed by Chang and Lin (2011). The classification algorithm using DAGSVM is summarized as follows:

1. Form the feature vectors by concatenating the spectral shape, spectral contrast and sub-band flux features.
2. Randomly select 5,000 feature vectors from a particular genre. Train the SVM for each decision node in the DDAG.
3. Construct the DDAG using the trained SVMs.
4. For a decision node, classify the test song features (1291 vectors for a 30-second clip). The preferred class (genre) of the test song is determined from the majority class of the classified features.
5. Proceed down to the next decision node.
6. Repeat steps 4 and 5.

5.1.3 Gaussian Mixture Models

This technique takes feature vectors associated to a given genre and uses them to derive a probability density function (PDF) that models the genre. Such a PDF results from a weighted combination of a number of Gaussian PDFs. A Gaussian mixture model (GMM), with M components where the contribution of the m-th component is weighted by a prior P_m with $P_m \geq 0$ and $\Sigma P_m = 1$, is given by:

$$p(x,\Theta)=\sum_{m=1}^{M}P_m\mathrm{N}(x\mid\mu_m,\Sigma_m) \tag{5.2}$$

where x is the observation, μ is the mean, Σ is the covariance matrix, Θ are the parameters that must be estimated per GMM. The parameters of the GMM are usually adjusted by means of an iterative process called Expectation Maximization (EM). The EM algorithm is iterative and converges relatively fast after a few iterations. Once trained, each GMM is used to estimate the probability that a feature vector was generated by the genre associated to that GMM. The final classification is given by the genre with the greatest probability. The initial estimates can be completely random, or can be computed using other algorithms such as k-means.

The EM algorithm consists of two steps. First, the expectation is computed. That is, the probability that an observation x_n was generated by the m-th component. Second, the expectation is maximized. That is, the parameters in Θ are updated based on the expectations. The expectation step is:

$$P(m\mid x_n,\Theta)=\frac{p(x_n\mid m,\Theta)P_m}{p(x_n)}=\frac{\mathrm{N}(x_n\mid\mu_m,\Sigma_m)P_m}{\sum_{m'=1}^{M}\mathrm{N}(x_{n'}\mid\mu_{m'},\Sigma_{m'})P_{m'}} \tag{5.3}$$

The maximization step is:

$$\begin{aligned}\mu_m^{*}&=\frac{\sum_n P(m\mid x_n,\Theta)x_n}{\sum_{n'}P(m\mid x_{n'},\Theta)}\\ \Sigma_m^{*}&=\frac{\sum_n P(m\mid x_n,\Theta)(x_n-\mu_m)(x_n-\mu_m)^T}{\sum_{n'}P(m\mid x_{n'},\Theta)}\\ P_m^{*}&=\frac{1}{N}\sum_n P(m\mid x_n,\Theta)\end{aligned} \tag{5.4}$$

For genre classification, we integrate the concept of Directed Acyclic Graph with GMM (DAGGMM) for faster comparisons. A pair of GMMs with full covariance matrix is used as the decision function for each node in the DDAG. Figure 5.4 shows the block diagram of the genre classifier using DAGGMM. We implement DAGGMM in MATLAB® with

the Netlab toolbox[37] developed by Ian Nabney and Christopher Bishop. The procedure for genre classification using DAGGMM is given below:

1. Form the feature vectors by concatenating the spectral shape, spectral contrast and sub-band flux features.
2. Randomly select 10,000 feature vectors from each genre and train the GMMs.
3. Construct the DDAG using the trained GMMs.
4. For a decision node, estimate the likelihood of the test song features (1291 vectors for a 30-second clip) under the pair of GMMs at that node.
5. Compare the values of the two probabilities for each feature vector, a feature is predicted to come from the GMM with the greater likelihood. The preferred class of the test song is determined from the majority of the predicted feature vectors.
6. Proceed down to the next decision node.
7. Repeat steps 4 to 6.

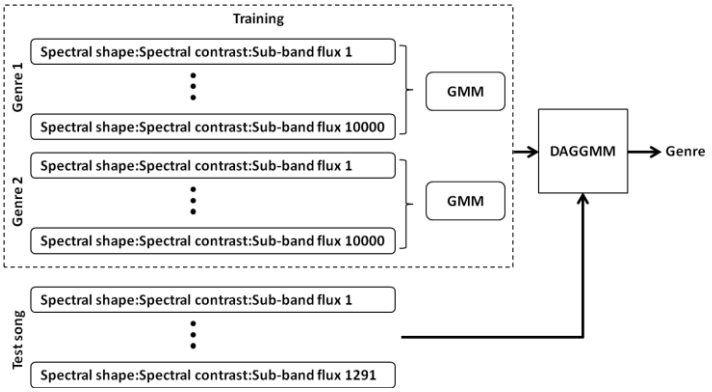


Figure 5.4: Block diagram of genre classification using DAG GMM.

5.2 Evaluation

In this section, the performance of k-NN, DAGSVM, and DAGGMM is evaluated using three datasets. Classification accuracy scores are averaged after 3-fold cross validation.

5.2.1 Datasets

Three datasets are used for local experiments. These are the DB-S, GTZAN and ISMIR 2004 datasets. The DB-S and GTZAN datasets have been used in timbre similarity experiments. These tracks are then divided into training (70%) and test (30%) sets such

[37] http://www.aston.ac.uk/eas/research/groups/ncrg/resources/netlab/

that the number of songs for each genre are almost equally distributed. The third dataset includes the training and testing sets for the ISMIR 2004 genre classification contest[38](“ISMIR 2004 Audio Description Contest-Genre/Artist ID Classification and Artist Similarity”)(“ISMIR 2004 Audio Description Contest-Genre/Artist ID Classification and Artist Similarity”)(“ISMIR 2004 Audio Description Contest-Genre/Artist ID Classification and Artist Similarity”)(“ISMIR 2004 Audio Description Contest-Genre/Artist ID Classification and Artist Similarity”)(“ISMIR 2004 Audio Description Contest-Genre/Artist ID Classification and Artist Similarity”)(“ISMIR 2004 Audio Description Contest-Genre/Artist ID Classification and Artist Similarity”)(“ISMIR 2004 Audio Description Contest-Genre/Artist ID Classification and Artist Similarity”)(“ISMIR 2004 Audio Description Contest-Genre/Artist ID Classification and Artist Similarity”)(“ISMIR 2004 Audio Description Contest-Genre/Artist ID Classification and Artist Similarity”)(“ISMIR 2004 Audio Description Contest-Genre/Artist ID Classification and Artist Similarity”)(“ISMIR 2004 Audio Description Contest-Genre/Artist ID Classification and Artist Similarity”)(“ISMIR 2004 Audio Description Contest-Genre/Artist ID Classification and Artist Similarity”)(“ISMIR 2004 Audio Description Contest-Genre/Artist ID Classification and Artist Similarity”). Table 5.2 lists the description of the datasets used in the experiments. The training set is used to create models whereas the testing set is used to derive statistics.

TABLE 5.2: Description of datasets used in genre classification experiments

Dataset	*#Train*	*#Test*	*#Classes*
DB-S	167	83	5
GTZAN	667	333	10
ISMIR 2004	729	729	6

[38] http://ismir2004.ismir.net/genre_contest/index.htm

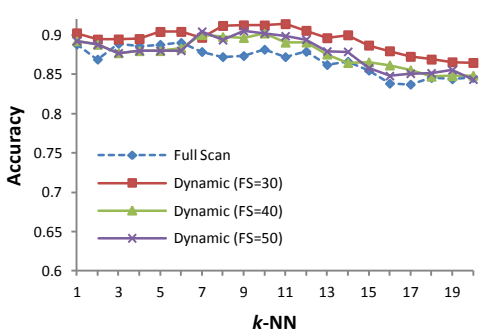
Three-fold cross-validation experiments are performed with the DB-S and GTZAN datasets. In three-fold cross validation, the original sample is randomly partitioned into three subsamples. Of the three subsamples, a single subsample is retained for testing the model, whereas the remaining two subsamples are used for training data. The cross-validation is then repeated three times, with each of the three subsamples used exactly once as the testing data. The three results from the folds are then averaged to produce single estimation. This process avoids *overfitting* of the models to the training data. Unless stated otherwise, all the statistics presented are averaged from three-fold cross-validation.

5.2.2 Results

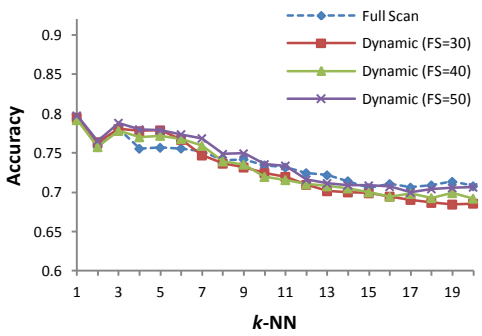
In the experiments, we compared three approaches (k-NN, DAGSVM, DAGGMM) using the features described in Section 2.3.2. In the initial experiment, we used the full scan and filter-and-refine timbre similarity systems to compute song-level features similarity. Using the ranking produced by these systems, the test song is labelled with the genre most prevalent among the k closest training songs. For these experiments k was varied from 1 to 20. The results are summarized in Figure 5.5.

The performance of this classifier is dependent on the timbre similarity algorithms. The best accuracies for the DB-S, GTZAN and ISMIR 2004 datasets are 0.91, 0.8, and 0.83 respectively. Thus, the high accuracies validate the performance of the similarity algorithms.

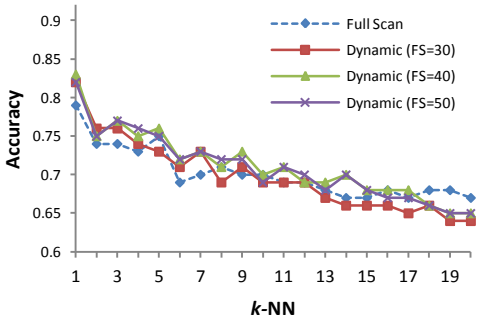
The general trend is that the classification accuracy decreases as the number of nearest neighbour (k) increases beyond the optimum value. For the DB-S dataset, the best accuracy is obtained when k=11 for the filter-and-refine system with a filter size=30. For the other datasets, the best accuracy occurs when k=1. This observation can be due to artist effect since the datasets contain songs from the same artists. There is a high probability that the first nearest neighbour to a test song will be a song by the same artist. Songs from the same artist are most likely tagged with the same genre, thereby making the first nearest neighbour as the best genre predictor.



(a) DB-S



(b) GTZAN



(c) ISMIR 2004

FIGURE 5.5: Genre classification results using k-NN.

In most cases, the filter-and-refine systems outperform the full scan system. For the DB-S and ISMIR 2004 datasets, the filter-and-refine system (filter size=30, k=1) exceeds the performance of the full scan system by 1.4 and 3 percentage points respectively. This means that its level of performance can be achieved at a fraction of the full scan system's runtime. In the next experiment, we compare the performance of the k-NN classifier (k=1) with DAGSVM and DAGGMM.

Table 5.3 shows the average classification accuracies of the different classifiers and datasets. For the DB-S dataset, the DAGSVM classifier performed best with an accuracy of 0.94. It is also the most consistent with a standard deviation of 0.06. For the GTZAN and ISMIR 2004 datasets, the k-NN classifiers outperformed DAGSVM. The k-NN classifier is also the most consistent for the GTZAN dataset with a standard deviation of 0.09. The results of the DAGGMM classifier are much lower compared to k-NN and DAGSVM, even when the number of Gaussian mixtures is varied.

TABLE 5.3: Classification accuracy and standard deviation

	k-NN Full Scan (k=1)	*k*-NN Dynamic (FS=30, *k*=1)	DAGGMM (5 comp)	DAGGMM (10 comp)	DAGGMM (15 comp)	DAGGMM (20 comp)	DAGSVM
DB-S							
pop rock	0.87	0.94	0.47	0.66	0.78	0.65	0.93
classical	1	1	0.9	0.89	0.9	0.9	0.98
electronica	0.6	0.64	0.9	0.9	0.94	0.92	0.84
hard rock	0.97	0.94	0.39	0.3	0.23	0.17	0.95
hip-hop	1	1	1	1	1	1	1
accuracy	0.89±0.17	0.9±0.15	0.73±0.28	0.75±0.28	0.77±0.31	0.73±0.34	**0.94±0.06**
GTZAN							
blues	0.91	0.91	0.51	0.51	0.56	0.58	0.97
classical	0.96	0.96	0.9	0.9	0.9	0.89	0.95
country	0.68	0.72	0.06	0.11	0.08	0.07	0.75
disco	0.73	0.76	0.1	0.04	0.1	0.1	0.58
hip-hop	0.77	0.72	0.63	0.58	0.71	0.71	0.68
jazz	0.85	0.83	0.55	0.53	0.6	0.61	0.76
metal	0.89	0.85	0.05	0.06	0.07	0.07	0.93
pop	0.8	0.8	1	1	1	1	0.96
reggae	0.62	0.65	0.41	0.48	0.49	0.48	0.6
rock	0.73	0.76	0	0	0	0	0.47
accuracy	0.79±0.11	**0.8±0.09**	0.42±0.36	0.42±0.36	0.45±0.37	0.45±0.37	0.77±0.18
ISMIR 2004							
classical	0.98	0.98	0.85	0.87	0.98	0.93	0.97
electronica	0.65	0.67	0.88	0.89	1	0.89	0.73
jazz/blues	1	1	0.54	0.54	0.81	0.54	0.88
metal/punk	0.78	0.84	0	0	1	0	0.82
rock/pop	0.68	0.74	0.17	0.11	0.25	0.21	0.62
world	0.65	0.69	0.27	0.25	0.26	0.2	0.61
accuracy	0.79±0.16	**0.82±0.15**	0.45±0.36	0.44±0.39	0.55±0.43	0.46±0.39	0.77±0.14

Focusing on the performance of the systems per genre, all the classifiers are effective for classical and hip-hop songs. The *k*-NN and DAGSVM classifiers are more effective than DAGGMM for pop rock, jazz, blues, and hard rock/metal songs. The *k*-NN classifier is the best classifier for general rock and world music; DAGGMM is the best classifier for electronica music.

The results show that SVM and *k*-NN have the advantage over GMM in terms of classification accuracy and consistency. The question now is which classifier is more practical for large datasets. Both SVM and GMM use 10,000 feature vectors for training,

but the training times for the SVM were considerably higher than GMM. Aside from training the SVM itself, the training time is compounded by the *grid search* that tries to find the best parameters (C and γ) for the SVM. For example, the training time of SVM is at least 200 seconds whereas the GMM takes less than 2 seconds. If we increased the number of training data, e.g. due to the addition of new songs in the database, then the training time would be longer. However, this should not be a problem since the training can be done offline.

In contrast, the k-NN classifier has no training phase. The computational complexity depends on estimating song-level similarities. Moreover, using the filter-and-refine method to find the nearest neighbours offers a practical solution to handle large databases.

5.3 MIREX 2013 audio classification task results

We submitted two algorithms for the MIREX 2013 audio classification task. Both systems use the k-NN based architecture as shown in Figure 5.1. The first algorithm (DM3) uses the full linear scan implementation to compute song-level similarity; the second algorithm (DM4) uses filter-and-refine implementation with dynamic filtering (filter size=50).

There are four sub-tasks for this evaluation: 1) US pop music genre classification, 2) classical composer identification, 3) Latin music genre classification, and 4) mood classification. However, we are only interested in the first two sub-tasks. The submitted algorithms were evaluated with 3-fold cross validation. Album filtering was used for classical composer identification and artist filtering was used for US pop music genre classification.

5.3.1 Current techniques

There were 14 submissions from 7 institutions for the 2013 audio classification task. The participants and their respective submission codes are tabulated in Table 5.4. The

features extracted and classifiers used are summarized in Table 5.5. Among all the submissions, our submissions are the only ones that used the k-nearest neighbour approach. There are three submissions (BBLJK, JJ, SSKS) that used SVM; AP used deep learning and DS used a Gaussian mixture model. The description for team CJ's submissions were not provided. The features used are related to timbre and rhythm.

Table 5.4: Participants for the 2013 MIREX audio classification task

Contributors	*Team code*	*Research Institution*
Aggelos Pikrakis	AP1	University of Piraeus
Byun, et al.	BBLJK1-BBLJK5	Sejong University, Korea Electronics Technology Institute
Pei-Hsuan Chou, Jyh-Shing Roger Jang	CJ1, CJ2	National Taiwan University
Franz de Leon, Kirk Martinez	DM3, DM4	University of Southampton
Dan Stowell	DS1	Queen Mary University of London
Ming-Ju Wu, Jyh-Shing Roger Jang	JJ1, JJ2	National Taiwan University
Seyerlehner, et al.	SSKS1	Johannes Kepler University

Table 5.5: Summary of the features and classifiers used by the submitted systems to the MIREX 2013 audio classification task

Team Code	*Features*	*Classifier*
AP1	Rhythm signatures based on self-similarity analysis	Deep learning network
BBLJK1-BBLJK5	Timbre features {MFCC, decorrelated filter banks, spectral contrast}, spectro-temporal features	SVM
CJ1,CJ2	NA	NA
DM3, DM4	Timbre features {spectral shape, spectral contrast, sub-band flux}	k-nearest neighbor
DS1	Timbre features {MFCC}	Gaussian mixture model
JJ1, JJ2	Visual features from spectrogram, acoustic features from MFCCs	SVM
SSKS1	Block-level timbre and rhythm features	SVM

5.3.2 Genre classification

For this task, the submitted systems should accurately classify music audio according to the genre of the track. The tracks in the dataset are drawn from MIREX's USPOP and USCRAP collections. The dataset contains 7000 tracks from 10 genres, with 700 clips from each genre. The genres include blues, jazz, country/western, baroque, classical, romantic, electronica, hip-hop, rock, and hard rock/metal.

Figure 5.6 compares the resulting average classification accuracies of the submitted systems. System JJ achieved the best performance with an accuracy of 0.76. This is followed by SSKS and BBLJK. These systems used SVM as classifier. Our submissions DM3 and DM4 achieved an average accuracy of 0.62 and 0.64 respectively. Similar to the results of the MIREX audio music similarity, the filter-and-refine system outperformed the full linear scan system.

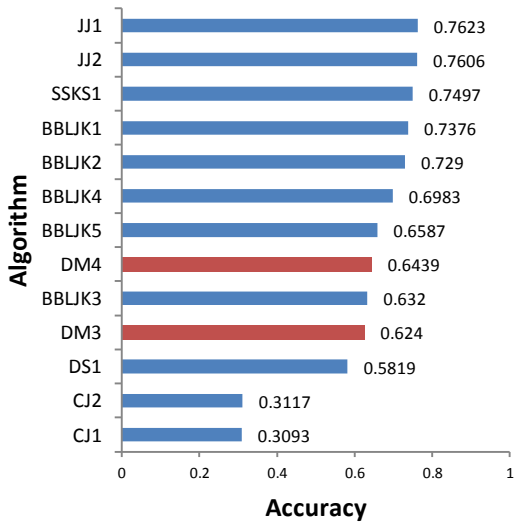


FIGURE 5.6: Genre classification accuracy of the submitted systems to MIREX 2013 genre classification task

The classification accuracies per genre of the submitted system are listed in Table 5.6. Suppose we set the minimum satisfactory level at 70%; the study by Gjerdingen and Perrott (2008) found that participants in a genre classification experiment agreed with the genres assigned by music companies 70% of the time. The results from our submissions are highlighted and the values that satisfy the satisfactory level are shown in bold. At this level, our submissions perform well with baroque, classical and rap/hip-hop genres. On the other hand, our submissions are least effective with edance and rockroll where the classification accuracies are below 50%. The best submission, JJ1, satisfies the satisfactory level for all genres except rockroll and romantic genres. All algorithms had classifying tracks from romantic genre.

TABLE 5.6: Classification accuracy per genre

Genre	*BBLJK1*	*BBLJK2*	*BBLJK3*	*BBLJK4*	*BBLJK5*	*CJ1*	*CJ2*	*DM3*	*DM4*	*DS1*	*JJ1*	*JJ2*	*SSKS1*
baroque	80.1	79.6	62.9	71.7	67.7	28.3	28.9	**79**	**81.9**	63.9	87.9	87.3	80.3
blues	72.9	72.1	61.7	74	70.9	26.1	25.3	66.9	67.3	63	78.1	78.1	80.4
classical	72.1	69	63.4	68.7	64.1	18.4	16.4	**78.9**	**79.7**	69	74.9	74.6	71.9
country	81	78.3	66.7	74.4	66.3	25	26.6	62.7	64.3	43	81.6	81.3	77.6
edance	75.3	75.1	70.3	71.3	67.4	28.7	28.4	39.7	45.3	54.1	76.9	76.4	76.7
jazz	75.6	76.1	60	66.9	60.3	26.7	27.3	67.9	65.9	49.6	80.1	79.9	82.3
metal	75.1	75.6	72.4	74.3	73.1	21.6	22.7	68.9	68.3	76.1	74.7	74.9	71.9
raphiphop	90.9	90.4	87.6	90.4	87.9	27.3	27.1	**78.6**	**80.4**	82	88	88	88
rockroll	53.3	54.3	39	48.1	45.4	88.3	89.4	33.4	39.1	45.1	57.1	57.1	55.6
romantic	61.3	58.4	48	58.4	55.6	18.9	19.6	48.1	51.7	36	63	63	65.1

The performance of our audio classifiers is highly correlated with the genre neighbourhood clustering accuracy reported in Table 4.7. This is expected since the audio classifiers use the same core timbre similarity algorithm. To understand the accuracy values, the classification confusion matrix of DM4 is shown in Table 5.7. It shows that confusions occur among overlapping genres: 1) baroque, classical, country; 2) blues, jazz; 3) country, metal, rockroll; and 4) edance, raphiphop. Among the four clusters, the first cluster is timbrally distinct from any other popular genres. Thus, our systems have sufficient intelligence to make successful predictions from these genres. The lowest classification accuracy (39%) is observed for rockroll. The rockroll songs are confused with country and metal genres which overlap naturally in terms of instrumentation. These logical confusions again validate that our systems find timbrally similar songs.

TABLE 5.7: Classification confusion matrix (DM4)

actual / predicted	baroque	blues	classical	country	edance	jazz	metal	raphiphop	rockroll	romantic
baroque	0.82	0.02	0.04	0.01	0.01	0.01	0.00	0.00	0.00	0.20
blues	0.01	0.67	0.00	0.06	0.01	0.15	0.00	0.00	0.03	0.01
classical	0.07	0.01	0.80	0.01	0.00	0.02	0.00	0.00	0.00	0.25
country	0.00	0.10	0.00	0.64	0.05	0.09	0.05	0.03	0.25	0.01
edance	0.00	0.00	0.00	0.01	0.45	0.01	0.04	0.09	0.03	0.01
jazz	0.01	0.14	0.01	0.09	0.03	0.66	0.01	0.01	0.04	0.01
metal	0.00	0.00	0.00	0.02	0.12	0.01	0.68	0.04	0.19	0.00
raphiphop	0.00	0.01	0.00	0.01	0.23	0.01	0.02	0.80	0.05	0.00
rockroll	0.00	0.02	0.00	0.14	0.09	0.02	0.19	0.02	0.39	0.00
romantic	0.08	0.02	0.15	0.00	0.00	0.01	0.00	0.00	0.00	0.52

5.3.3 Audio classical composer

For this task, the algorithms should classify 30-second music clips according to the composer of the track. The dataset used include 11 classical composers, 252 clips per composer. There are three composers from the baroque era: Bach, Handel, Vivaldi; and there are eight composers from the classical/romantic era: Beethoven, Brahms, Chopin, Dvorak, Haydn, Mendelssohn, Mozart, Schubert. Figure 5.7 shows the classification accuracies of the submitted systems. The results show that our systems DM4 and DM3 are ranked first and third place respectively. This is unexpected since the genre classification results show that the SVM classifiers outperform our k-NN systems. This means that our proposed feature spaces and timbre similarity estimation are state-of-the-art methods, at least within the bounds of the classical music genre.

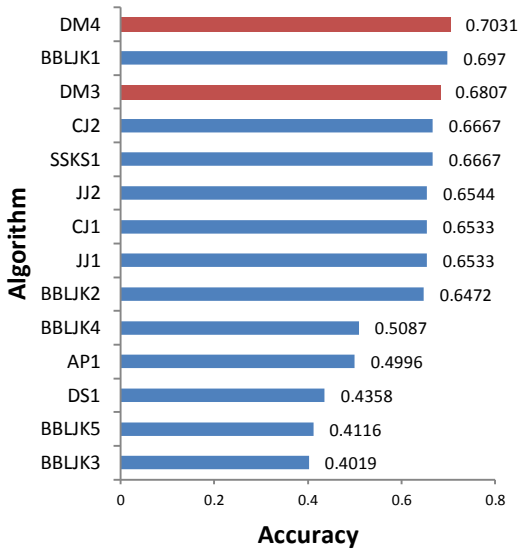


FIGURE 5.7: Classical composer classification accuracy of the submitted systems to MIREX 2013 genre classification task

5.4 Summary

In this chapter, we applied the concept of timbre similarity to automatic genre classification through k-nearest neighbour method. The idea is to classify an untagged song with the genre of the most timbrally similar song. We compared this approach with support vector machine and Gaussian mixture model. The results from our local

experiments show that the performance of k-NN is comparable with DAGSVM over three datasets, and significantly better than DAGGMM.

We submitted two k-NN based systems to the MIREX audio classification task. The first system uses the full scan similarity estimation while the second uses the filter-and-refine method with dynamic filtering. The MIREX results show that the filter-and-refine method outperformed the full linear scan system. Based on the genre classification confusion matrix and classical composer classification accuracy, our systems are state-of-the-art methods in the classical music genre. The MIREX results also show that the genre classification accuracies are pulled down by classification confusions of timbrally similar genres.

Chapter 6: Conclusions

This thesis has presented the development of a content based music retrieval system using polyphonic timbre similarity. This system can be useful in developing applications for digital music collections to perform music discovery, recommendation, and classification. This chapter brings together the important points of preceding chapters, including the system limitations, as well as suggest ideas for further work based on insights gained.

6.1 Summary and Conclusions

Music Information Retrieval is a dynamic research field that encompasses several disciplines. This diversity was described in terms of the variety of tasks involved in the annual Music Information Retrieval Evaluation eXchange (MIREX). This work focused on developing an effective and efficient content based audio retrieval system using polyphonic timbre similarity.

Several methods for music similarity estimation found in literature were enumerated in Chapter 2. Such application becomes increasingly important in the digital music landscape. Content-based methods serve to complement other music similarity approaches that require large manpower resources. We also presented the basic components of a content-based retrieval system. Emphasis was given to the feature extraction process. Common low-level and high-level audio descriptors were discussed highlighting the semantic gap that exists between them. This semantic gap is an issue that must be addressed to make content-based methods more relevant to users.

The perception of audio similarity is dependent on a user's preference, cultural, and musical background. Hence, we removed this subjective factor by focusing on polyphonic timbre similarity, i.e. having similar sound texture. The main objective is to determine a quantitative model of polyphonic timbre by applying signal processing techniques.

We discussed the complexities of timbre modelling. Unlike pitch or loudness that has direct correlations to fundamental frequency and sound energy, respectively, timbre is multidimensional and there is no single feature that captures all its complexities. In several studies, the spectral envelope, represented by the Mel-frequency cepstral coefficients (MFCCs), has been the main descriptor of timbre. Five timbre attributes were identified by Schouten (1968) and one of these is spectral envelope. In our research, we also considered two other attributes which are: 1) the range between tonal and noise-like character, and 2) the temporal unfolding and shaping of sound spectra. The two remaining timbre attributes requires identifying and decomposing a sound to its individual instruments which is beyond the scope of our study.

The process of extracting timbre features from an audio signal is described. It begins by normalizing the audio signal, then segmenting it into small frames. The segmented signal is assumed to be pseudo-stationary, i.e. its statistics are constant or at least slowly-varying. The time-domain signal is transformed to frequency domain through Fourier transform. In the frequency domain, we extract MFCCs (Davis and Mermelstein 1980), spectral contrast coefficients (Jiang et al. 2002), and sub-band flux coefficients (Alluri and Toiviainen 2009). These three feature spaces are motivated by the timbre attributes discussed in Section 2.3.1. In addition, we extracted a fourth feature space, which is a set of spectral distribution descriptors that may serve as an alternative feature space for spectral envelope.

Different ways of summarizing the four feature spaces were also enumerated. Based on literature, the bag-of-frames framework is suitable for our low-specificity application. The feature vectors computed for every frame from a song are modelled as single multivariate Gaussian distributions. This model only requires computing the mean and covariance matrix of the feature vectors. A closed form solution of the Kullback-Leibler (KL) divergence (Penny 2001) exists that can be used to estimate similarity between two Gaussian models. We then applied a mathematical transformation proposed by Charbuillet et al. (2010) to turn KL divergence into a metric distance value.

In Chapter 3, we presented the evaluation results of the four feature spaces over four datasets. The main performance measure used was precision at n items. We optimized each feature space by tuning relevant variables. The performance of MFCCs served as our baseline. Appending the MFCCs with its time derivatives did not result in any significant improvements. However, appending the MFCCs with the spectral distribution descriptors resulted in minor improvements in precisions compared to the baseline. We define this new feature space as spectral shape. The precision results from spectral shape, spectral contrast, and sub-band flux showed that each feature space is effective in particular genres.

We also visualized a 3-d polyphonic timbre space from the three feature spaces. After mapping the songs from the DB-S dataset, we observed that classical and hip-hop songs formed noticeable clusters. This led us to linearly combine the normalized distance matrices of the feature spaces. The combination of features resulted in a significant increase in precisions compared to precisions from any individual feature space. It also has the advantage of reducing the occurrence of hub and orphan songs.

We discussed our approach to make our audio retrieval system more efficient in Chapter 4. We set the acceptable response time of our system to be less than 10 seconds. The computational complexity of a full linear scan system using combined features does not satisfy our target response time, even with a small dataset. To lessen computational complexity, we adopted the filter-and-refine method. The idea in the filter step is to quickly return a set of candidate songs to a query song. We used a modified FastMap algorithm (Faloutsos and Lin 1995) to embed the features in Euclidean space. Hence, similarity estimation can be done much faster using Euclidean distance. We proposed two variations of the filter step: 1) static filtering that returns a fixed number of candidate songs, and 2) dynamic filtering that returns a variable number of candidate songs.

The refine step computes the exact similarity estimate between the query song and candidate songs using KL divergence. Our evaluation results show the filter-and-refine method with dynamic filtering has the best potential in terms of performance, response time, and the ability to reduce hub and orphan songs.

We validated the performance of our full linear scan and filter-and-refine systems by submitting to the MIREX 2013 audio music similarity task. Based on the neighbourhood clustering statistics with respect to artist and album matches, our submissions are comparable with the best performing systems. Our filter-and-refine system with dynamic filtering got the highest precision at 5 items according to artist and album matches. The results confirm that our systems can return timbrally similar songs. The MIREX results also proved that the architecture of our systems minimizes hub and orphan songs.

We also applied our timbre similarity systems to automatic genre classification. An untagged song is classified with the label of the estimated nearest song. Our local experiments showed that the classification accuracy of the nearest neighbour approach is comparable with the result using support vector machine. We submitted our systems to the MIREX 2013 audio classification task. Based on the genre classification confusion matrix and classical composer classification accuracy, our systems are state-of-the-art methods in classical music genre. The MIREX results also show that our systems make confusions between timbrally similar genres.

A list of contributions to music information retrieval made by this thesis was enumerated in the introduction. The novel contributions of this study are reiterated here:

- Improvement of the spectral envelope feature space by appending spectral distribution descriptors
- Development of a music similarity system using three feature spaces (spectral shape, spectral contrast, sub-band flux) based on attributes of timbre
- Demonstration of a 3-d polyphonic timbre space where each dimension corresponds to an attribute of timbre
- Development of a linear combination technique for combining the feature spaces
- Development of an efficient retrieval system by adopting the filter-and-refine method, including two novel variations of the filtering step
- Demonstration of a technique for audio classification by propagating the nearest neighbor's label

6.2 Limitations

Pure content-based audio similarity methods are still far from the way humans evaluate audio similarity. Although this process seems so effortless for human listeners, it is a very complex process involving personal and cultural aspects. On the other hand, some applications may tolerate a certain variance for automatically retrieved songs. However, it becomes increasingly important to incorporate the end user's personal preference or listening behaviour to make these applications more meaningful.

Another important limitation of content-based methods is that they cannot characterize the quality of a music track. For example, they cannot distinguish between a novice and a seasoned artist doing a jazz piece. In a similar manner, all the systems submitted to the MIREX audio music similarity task are not designed to recognize a query song's genre. Rather, the systems are designed to estimate the similarity of features in the hope that the nearest songs belong to the same genre. Genre similarity is more than just timbre and rhythm similarity. Genres exist as a group of stylistic tendencies, codes, conventions, and expectations that become meaningful in relation to one another at a particular moment in time (Hesmondhalgh and Negus 2002).

Figure 6.1 plots the precision at 5 items of the best submission to MIREX audio music similarity and retrieval task. The plot shows that from 2009 to 2010, the precision increased by 0.12. Since then, no significant improvements are observed despite the increasing complexities of the submissions every year. A glass ceiling still exists which proves the limitation of pure content-based retrieval systems. An option to further improve the performance of the systems is to combine them with additional information. Such information could be based on collaborative filtering data, or web-based community tags.

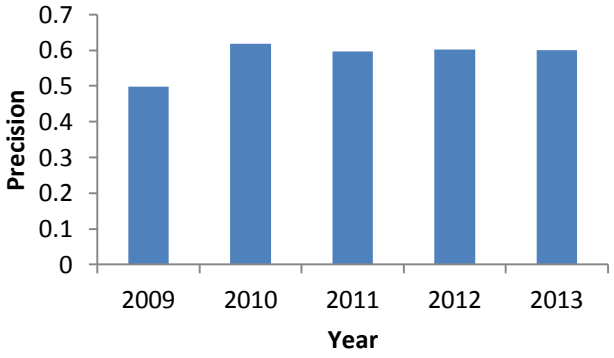


Figure 6.1: Precision at 5 items of the best submission to MIREX audio music similarity and retrieval task

6.3 Further Work

The MIR techniques described in this work have a lot of room for improvement. Comparing the systems developed in this study with the state-of-the-art, it can be said that the main difference is that the state-of-the-art derives more features. In fact, most systems exploit information from the magnitude spectrum. The best submission to the MIREX genre classification task even went as far as extracting visual features from a signal's spectrogram (Jang 2011). Hence, one direction for future work is to review other features and determine an efficient way to integrate these to our system. The architecture of our system is flexible enough to accommodate new feature spaces that may complement the current features used.

We can also learn from other MIR tasks. For example, the automatic music transcription task involves instrument recognition and onset detection. However, this may take more time as the performance of automatic transcribers is significantly below that of a human expert (Benetos et al. 2013). Perhaps a practical scenario is to have multiple systems or algorithms performing complementary tasks, e.g. onset detection, tempo estimation, key estimation, etc., and aggregate the musical information. In the long run, the main goal is to narrow the semantic gap between the low-level features and high-level concepts.

We have seen that the performance of content-based systems seem to have approached a limit or glass ceiling. It is suggested by Humphrey et al. (2013) that widely used hand-crafted features such as MFCCs are not sustainable as it paves the way for shallow and

limited architectures that usually fail to capture long–term musical structures . The authors are pushing for a paradigm shift that involves deep learning, a promising technique that has been developed by the machine learning community. The idea is seconded by Aucouturier and Bigand (2013). The authors believe that the MIR community should also consider the field of music cognition and neuroscience. This provides new research opportunities and may be mutually beneficial to both fields.

Finally, most of the MIR systems and techniques are evaluated over popular Western types of music. It would be interesting to expand their applications to folk or ethnic music. The author plans to pursue this research over a Philippine indigenous music collection. The MIR techniques have the potential to contribute in ethnomusicology research. It is hoped that applications will be developed to create a more interactive access to ethnic music and make the listeners appreciate the rich diversity of world music.

Appendix 1 MFCC Feature Extraction

These steps, starting with the transformation of the audio signal to the frequency domain are described in the following paragraphs.

Power Spectrum

Transforming the audio signal from the time-domain to the frequency-domain is important because the human auditory system applies a similar transformation. In particular, in the *cochlea* (which is a part of the inner ear) different nerves respond to different frequencies (Fastl and Zwicker 2006).

First, the signal is divided into short overlapping segments (e.g. 23ms and 50% overlap). The length of the segment ensures that the segmented signal is pseudo-stationary while the hop size keeps the continuity of the segments. Second, a window function (e.g. Hann window) is applied to each segment. This is necessary to reduce spectral leakage. Third, the power spectrum matrix P is computed using a Fast Fourier Transformation. Note that the signal is rescaled to have a maximum value of 96 dB since this is dynamic range (DR) of a 16-bit encoded music, i.e. DR=$20\log_{10}(2^{16})$≈96 dB. Figure 1 illustrates a segment before and after applying the window function, and the corresponding power spectrum.

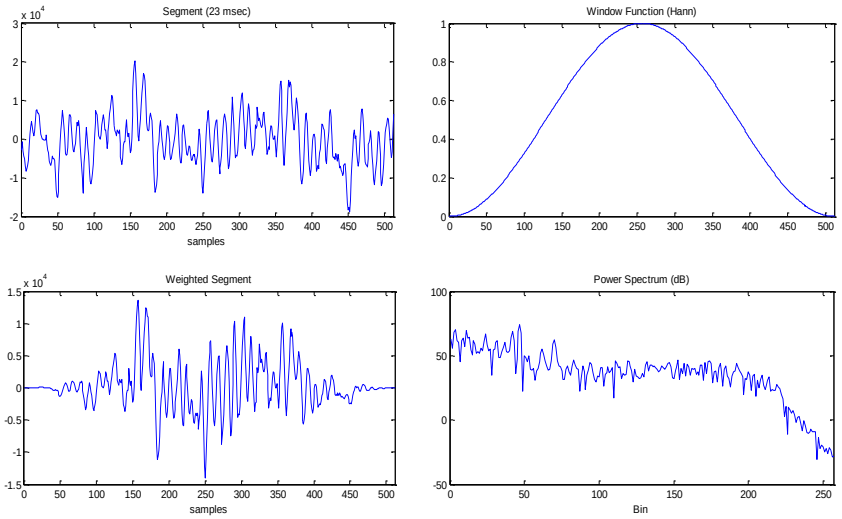


FIGURE 1: Visualization of power spectrum computation of 23 ms audio frame

Mel-frequency

The Mel-scale (Stevens 1937) is approximately linear for low frequencies (<500Hz), and logarithmic for higher frequencies. The reference point to the linear frequency scale is a 1000Hz tone which is defined as 1000 Mel. A tone with a pitch perceived twice as high is defined to have 2000 Mel, a tone perceived half as high is defined to have 500 Mel and so forth. The Mel-scale is defined as

$$m_{Mel} = 1127.01048 \log(1 + f_{Hz} / 700)$$

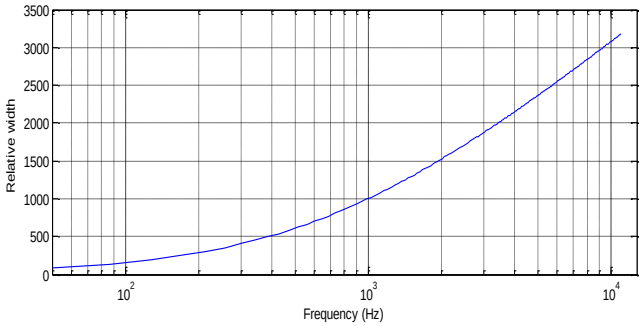


FIGURE 2: Relative width of the frequency bands according to the Mel-scale

The power spectrum is transformed to the Mel-scale using a filter bank consisting of triangular filters. Each triangular filter defines the response of one frequency band and is normalized such that the sum of weights for each triangle is the same. In particular, the height of each triangle is 2/d where d is the width of the frequency band. The triangles overlap each other such that the centre frequency of one triangle is the starting point for the next triangle, and the end point of the previous triangle (see Figure 3). The code used is based on Malcolm Slanley's Auditory Toolbox[39]. The Mel filter matrix and its effect on the power spectrum are shown in Figure 4. It can be observed that the spectrum is smoothed. However, information in the higher frequencies is lost due to lower resolution at higher frequencies.

[39] http://cobweb.ecn.purdue.edu/~malcolm/interval/1998-010/

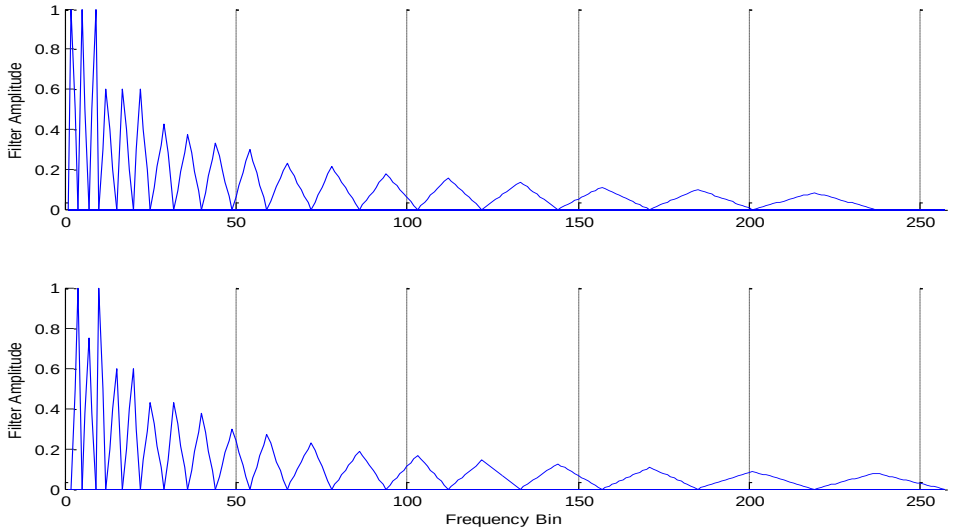


FIGURE 3: Triangular filters used to transform the power spectrum to the Mel-scale using 36 (Mel-spaced) frequency bands. The upper plot shows the *odd* (id=1,3,5…35) triangles, the lower shows the even triangles (id=2,4,6…36)

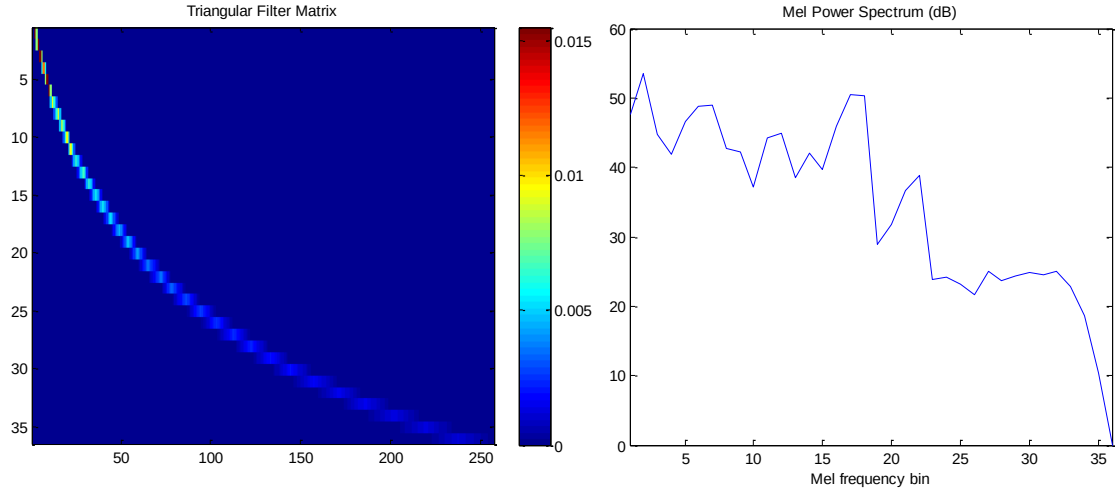


FIGURE 4: Mel filter matrix for and Mel power spectrum of the audio signal in Figure 1. The dimensions of the filter matrix are Mel frequency bands on the y-axis and FFT frequency bins on the x-axis. The dimensions of the Mel power spectrum are dB on the y-axis and Mel on the x-axis.

Decibel

Similarly to the non-linear perception of frequency, the human auditory system does not perceive loudness linearly (with respect to the physical properties of the audio signal). In particular, the just noticeable difference in loudness for sounds with a low intensity (Watts/m^2) is much smaller than for sounds with a high intensity.

A useful approximation of the loudness perception is to use a logarithmic ratio scale known as Decibel (dB). The important part of this scale is the reference to which the ratio is computed. In the examples used in this report the reference is 1, and the audio signals are rescaled (such that the maximum is 96 dB) with respect to this reference. This reference is the dynamic range of a 16-bit encoded music.

Discrete Cosine Transform

The Discrete Cosine Transform is applied to compress the Mel power spectrum. In particular, frequency bands (e.g. 36) are represented by fewer coefficients (e.g. 20). A side effect of the compression is that the spectrum is smoothed along the frequency axis which can be interpreted as a simple approximation of the spectral masking in the human auditory system; that is sounds of adjacent frequencies (pitches) may be heard as a combination tone rather than distinct sounds. Each of the rows of the DCT matrix corresponds to an eigenvector, starting with the most important one (highest eigenvalue) in the first row. The first eigenvector describes the mean of the spectrum. The second describes a spectral pattern with high energy in the lower half of the frequencies and low energy in the upper half. The eigenvectors are orthogonal. The DCT is applied to the Mel power spectrum by multiplying the DCT matrix to each segment. The effects of the DCT matrix on the Mel power spectrum are shown in Figure 5.

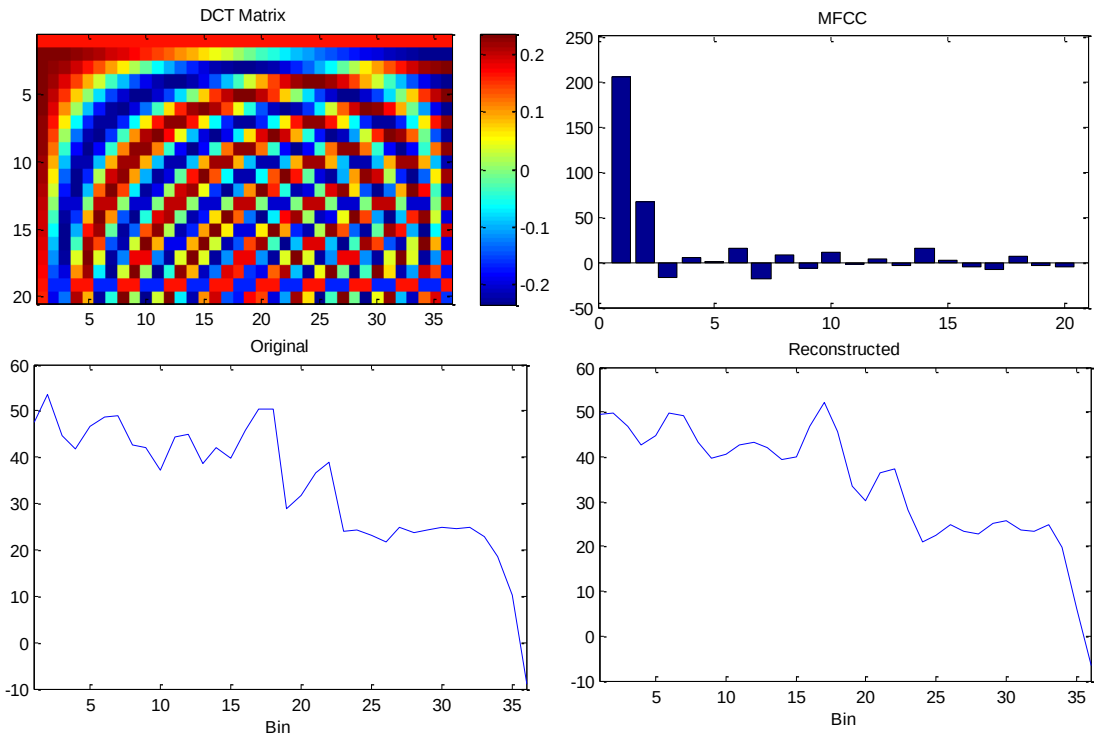


FIGURE 5: DCT matrix, MFCCs, and comparison of original and reconstructed Mel power spectrums (dB) for the audio signal used in Figure 1. The dimensions of the DCT matrix are DCT coefficients on the y-axis and Mel frequency bands on the x-axis.

The resulting MFCCs are a compressed representation of the original data. In particular, while the original audio signal has 512 samples per 23ms segment (22050Hz, mono) the MFCC representation only requires 20 values for 12msec (using 50% overlap for the power spectrum). Depending on the application the number of coefficients can vary. However, the number of coefficients is always lower than the number of Mel frequency bands used. To understand the smoothing effects of the DCT it is useful to look at the reconstructed Mel power spectrum and compare it to the original Mel power spectrum Figure 5.

Figure 6 illustrates the computation steps on a test music file (10 second sequence). Noticeable are (1) the changes in frequency resolution when transforming the power spectrum to the Mel power spectrum, (2) that MFCCs when viewed directly are difficult to interpret and that most of the variations occur in the lower coefficients, (3) the effects of the DCT-based smoothing (when comparing the Mel power spectrum with the reconstructed version).

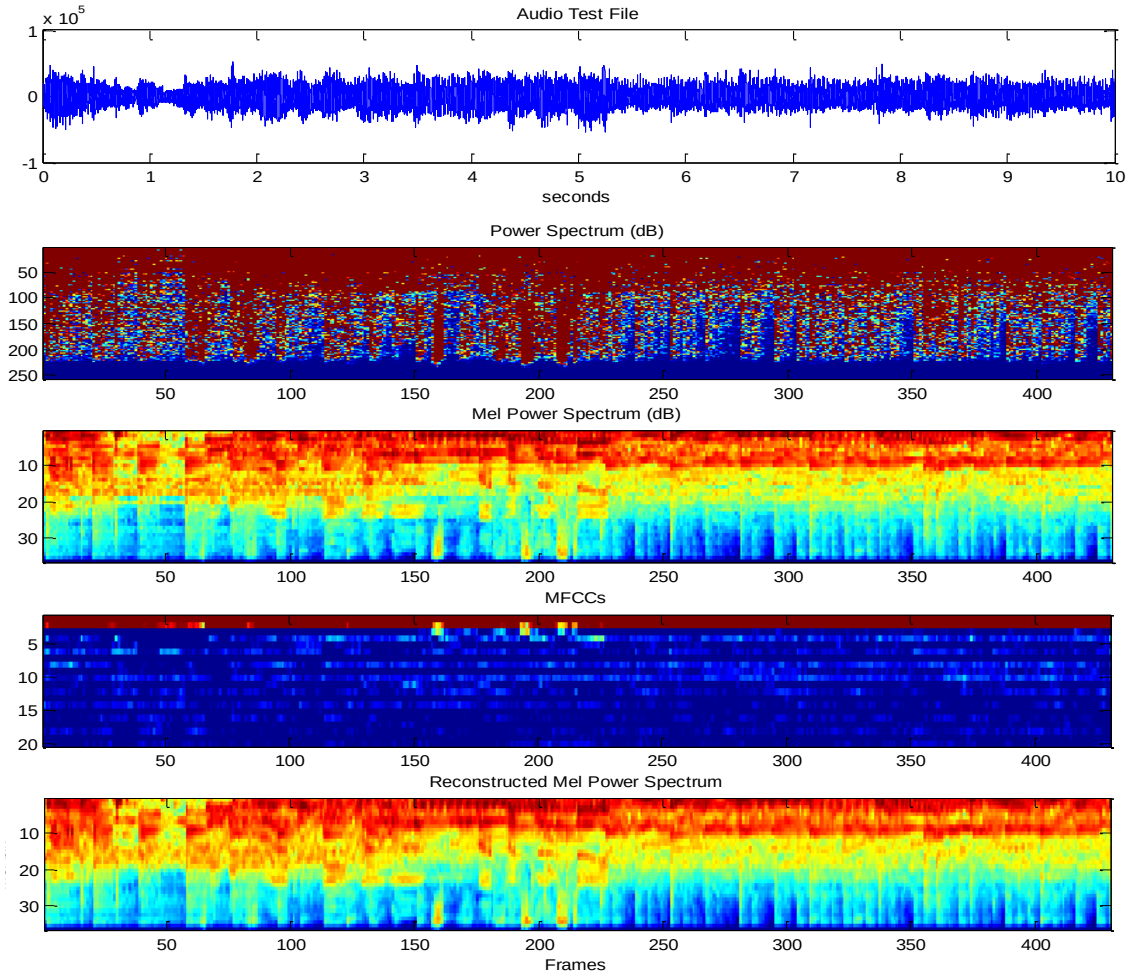


Figure 6: MFCC computation steps for a test music file. Time is plotted on the x-axis (the temporal resolution is 256/22050 seconds which is about 12msec per frame).

Appendix 2 MATLAB scripts

List of functions and sub functions

1. ExtractFeatures - extracts timbre features
    a. mydisp – creates log file
    b. ExtractSpectral – extracts MFCCs and spectral distribution descriptors
    c. ExtractSBF – extracts sub-band flux coefficients
    d. ExtractSC – extracts modified spectral contrast coefficients
2. ComputeDistance – computes estimated timbre similarity
    a. mydisp – creates log file
    b. ComputeSpectralDistance – computes spectral shape similarity
    c. ComputeSBFDistance – computes sub-band flux similarity
    d. ComputeSCDistance – computes spectral contrast similarity
    e. Norm_dist_mat – normalizes a distance matrix
3. FastMap – maps Gaussian features to k-dimensional Euclidean space using FastMap algorithm

```
function ExtractFeatures(out_dir,in_file)
%%
%% USAGE EXAMPLES
%%       ExtractFeatures(out_dir,in_file)
%%       ExtractFeatures('mypath/myOutputDirectory','mypath/myInputFile.txt')
%%
%% INPUT ARGUMENTS
%%
%% out_dir: directory to which this function can write output. (logfiles,
%%          and extracted features), files written to out_dir must be read
%%          by function "ComputeDistance"
%% in_file: path fo file containing list of wav files to extract features from
%%          all files are 22050Hz mono wav (this function checks if this is true)
%%
%% in_file format (text):
%%      path/to/audiofile/0000001.wav
%%      path/to/audiofile/0000002.wav
%%      path/to/audiofile/0000003.wav
%%      ...
%%      path/to/audiofile/9999999.wav
%%

%% HARDCODED PARAMETERS
data.submission_name = 'DM1';
c.fs = 22050; %% all files expected to be 22050 Hz


if nargin~=2,
    error('Number of input arguments is not 2. (try "help ExtractFeatures")')
end

if out_dir(end)~='/' && out_dir(end)~='\',
    out_dir(end+1)='/';
end
output_file = [out_dir,'Audio_features.mat'];

%% TEST WRITE ACCESS TO OUTPUT DIRECTORY
fid = fopen([out_dir,'testwritefile'],'w');
if fid==-1, error('cannot write to output directory'); end
fclose(fid);
delete([out_dir,'testwritefile']);

%% TEST READ ACCESS TO INPUT
fid = fopen(in_file,'r');
if fid==-1, error('cannot read from input file (does it exist?)'); end
fclose(fid);

%% START LOGFILE
logfile = [out_dir,'ExtractFeatures-',data.submission_name,'-logfile.txt'];
fid = fopen(logfile,'a');
if fid==-1, error('can''t append logfile'); end
fclose(fid);

mydisp(logfile,datestr(now));
mydisp(logfile, '-> ExtractFeatures called.')
mydisp(logfile,['   Output directory: ',out_dir])
mydisp(logfile,['   Input file: ',in_file])

%% LOAD INPUT FILE
data.filenames = textread(in_file,'%s','delimiter','\n');
```

```
if isempty(data.filenames),
    mydisp(logfile,'-- length(data.filenames)==0');
    mydisp(logfile,'-- something is wrong with the input file, or something went wrong reading from it.');
    error('-- test failed')
end
if isempty(data.filenames{end}), %% ignore empty lines in input file (e.g. at the end of the file)
    tmp = {};
    for i=1:length(data.filenames)-1,
        if ~isempty(data.filenames{i}),
            tmp{end+1} = data.filenames{i};
        end
    end
    data.filenames = tmp;
end
mydisp(logfile,['found ',num2str(length(data.filenames)),' files']);

% Sub band flux constants
%design and create 10 octave band filter
d=fdesign.lowpass('N,F3dB',4,25,22050);
Hd1=design(d,'butter'); %IIR butterworth design
d=fdesign.bandpass('N,F3dB1,F3dB2',4,25,50,22050);
Hd2=design(d,'butter'); %IIR butterworth design
d=fdesign.bandpass('N,F3dB1,F3dB2',4,50,100,22050);
Hd3=design(d,'butter'); %IIR butterworth design
d=fdesign.bandpass('N,F3dB1,F3dB2',4,100,200,22050);
Hd4=design(d,'butter'); %IIR butterworth design
d=fdesign.bandpass('N,F3dB1,F3dB2',4,200,400,22050);
Hd5=design(d,'butter'); %IIR butterworth design
d=fdesign.bandpass('N,F3dB1,F3dB2',4,400,800,22050);
Hd6=design(d,'butter'); %IIR butterworth design
d=fdesign.bandpass('N,F3dB1,F3dB2',4,800,1600,22050);
Hd7=design(d,'butter'); %IIR butterworth design
d=fdesign.bandpass('N,F3dB1,F3dB2',4,1600,3200,22050);
Hd8=design(d,'butter'); %IIR butterworth design
d=fdesign.bandpass('N,F3dB1,F3dB2',4,3200,6400,22050);
Hd9=design(d,'butter'); %IIR butterworth design
d=fdesign.highpass('N,F3dB',4,6400,22050);
Hd10=design(d,'butter'); %IIR butterworth design
Hd = [Hd1 Hd2 Hd3 Hd4 Hd5 Hd6 Hd7 Hd8 Hd9 Hd10];
clear d Hd1 Hd2 Hd3 Hd4 Hd5 Hd6 Hd7 Hd8 Hd9 Hd10

     %% CONSTANTS FOR FEATURE EXTRACTION
        o = zeros(length(data.filenames),1);

        data.feat.aggre.m           = zeros(length(data.filenames),29);
        data.feat.aggre.co          = zeros(length(data.filenames)*29,29);
        data.feat.aggre.ico         = zeros(length(data.filenames)*29,29);
        data.feat.glc.aggre_ico     = o;

        data.feat.flux.m            = zeros(length(data.filenames),10);
        data.feat.flux.co           = zeros(length(data.filenames)*10,10);
        data.feat.flux.ico          = zeros(length(data.filenames)*10,10);
        data.feat.glc.flx_ico       = o;

        data.feat.sc.m              = zeros(length(data.filenames),20);
        data.feat.sc.co             = zeros(length(data.filenames)*20,20);
        data.feat.sc.ico            = zeros(length(data.filenames)*20,20);
        data.feat.glc.sc_ico        = o;


    t0 = cputime;
    for i_files=1:length(data.filenames),
        t2 = cputime;

        %% START FEATURE EXTRACTION CODE

        seg1 = (i_files-1)*29+1:i_files*29;
        seg2 = (i_files-1)*10+1:i_files*10;
        seg3 = (i_files-1)*20+1:i_files*20;


        %signal preprocessing
        siz = wavread(data.filenames{i_files},'size'); %% use (at most) 30 sec from center

        if siz(1)>c.fs*30,
            x0 = ceil(siz(1)/2-c.fs*15);
            x1 = floor(siz(1)/2+c.fs*15);
        else %% use whatever is available
            x0 = 1;
            x1 = siz(1);
        end

        wav = wavread(data.filenames{i_files},[x0 x1]);

        wav = (wav - mean(wav)) / max(abs((wav - mean(wav)))); % normalize
        %signal

        if max(isnan(wav))==1 %considers the case of absolute silent audio
            wav = wavread(data.filenames{i_files},[x0 x1]);
            wav = wav+rand(x1,x0)-0.5;
            wav = (wav - mean(wav)) / max(abs((wav - mean(wav))));
        end


         %extract MFCC and spectral features
            [feat_m,feat_co,feat_ico,feat_max_ico] = ExtractSpectral(wav);
            data.feat.aggre.m(i_files,:)        = feat_m;
            data.feat.aggre.co(seg1,:)      = feat_co;
            data.feat.aggre.ico(seg1,:)     = feat_ico;
            data.feat.glc.aggre_ico(i_files)    = feat_max_ico;
            mydisp(logfile,['Extracting MFCC and spectral features on file ',num2str(i_files),'/',...
                num2str(length(data.filenames)),'...done']);
```

```
        clear feat_m feat_co feat_ico feat_max_ico


         %compute sub-band flux coefficients
            [feat_m,feat_co,feat_ico,feat_max_ico] = ExtractSBF(wav,Hd);
            data.feat.flux.m(i_files,:)     = feat_m;
            data.feat.flux.co(seg2,:)  = feat_co;
            data.feat.flux.ico(seg2,:) = feat_ico;
            data.feat.g1c.flx_ico(i_files)  = feat_max_ico;
            mydisp(logfile,['Extracting sub-band flux coeffs on file ',num2str(i_files),'/',...
                num2str(length(data.filenames)),'...done']);

        clear feat_m feat_co feat_ico feat_max_ico

         %compute spectral contrast
            [feat_m,feat_co,feat_ico,feat_max_ico] = ExtractSC(wav);
            data.feat.sc.m(i_files,:)       = feat_m;
            data.feat.sc.co(seg3,:)    = feat_co;
            data.feat.sc.ico(seg3,:)   = feat_ico;
            data.feat.g1c.sc_ico(i_files)   = feat_max_ico;
            mydisp(logfile,['Extracting spectral contrast coeffs on file ',num2str(i_files),'/',...
                num2str(length(data.filenames)),'...done']);

        clear feat_m feat_co feat_ico feat_max_ico

        if i_files>1,
            time.avg_t_sofar = (t2-t0)/(i_files-1);
            time.est_tot = (t2-t0)/(i_files-1)*length(data.filenames);
            time.est_rem = time.est_tot - (t2-t0);
            mydisp(logfile,[num2str(i_files),'/',num2str(length(data.filenames)), ...
                ' FeatExt est rem ',num2str(time.est_rem/60),'m, est tot ', ...
                num2str(time.est_tot/60),'m'])
            mydisp(logfile,['Extracting features on file ',num2str(i_files),'/',num2str(length(data.filenames)),...
                '...done']);
        end

   end
    tot_time_FE = cputime-t0;
    mydisp(logfile,['Total CPU Time [h]: ',num2str(tot_time_FE/60/60)])

    mydisp(logfile,['saving data to output file: ',output_file])
    a = version;
    if a(1)=='7', %% matlab 7 has default compression
        mydisp(logfile,'   Matlab 7')
        save(output_file,'data','-v6'); %% dont use compression
    else
        mydisp(logfile,'   Not Matlab 7')
        save(output_file,'data');
    end


    mydisp(logfile,'done. exiting ...')
    mydisp(logfile,datestr(now))

end

function mydisp(logfile,str) %% appends logfile
fid = fopen(logfile,'a'); disp(str); fprintf(fid,'%s\r\n',str); fclose(fid);
end

function [feat_m,feat_co,feat_ico,feat_max_ico] = ExtractSpectral(wav)
%%
%% extracts mfcc and spectral features
%%

%% CONSTANTS FOR FEATURE EXTRACTION
        wav_temp            = wav;

        c.fs                = 22050;
        c.num_mfcc_coeffs   = 20;  % number of mfcc coeffs
        c.num_mfcc_filt     = 36;  % number of Mel frequency bands for mfcc

        c.num_filt          = 19;  % number of Mel frequency bands for SC
        c.alpha             = 0.2; % defines number of samples for SC
        c.num_sc_coeffs     = 19;

        c.num_flux_filt     = 10;  % number of Mel frequency bands for flux
        c.num_flux_coeffs   = 10;  % number of flux coeffs

        c.seg_size          = 512; % 23ms if c.fs == 22050Hz
        c.hop_size          = 512;
        c.norm_freq         = linspace(0,c.fs,c.seg_size)/c.fs;

        f = linspace(0,c.fs/2,c.seg_size/2+1); %% frequency bins of P
        mel = log(1+f/700)*1127.01048; %orig mel scale

        c.kappa             = 0.85;
        c.frames            = 0.7; %percentage of frames to discarded

        % MFCC constants
        mel_idx = linspace(0,mel(end),c.num_mfcc_filt+2);

        f_idx = zeros(c.num_mfcc_filt+2,1);
        for i=1:c.num_mfcc_filt+2,
            [~, f_idx(i)] = min(abs(mel - mel_idx(i)));
        end
        freqs = f(f_idx);

        % height of triangles
        h = 2./(freqs(3:c.num_mfcc_filt+2)-freqs(1:c.num_mfcc_filt));

        c.mel_filter = zeros(c.num_mfcc_filt,c.seg_size/2+1);
        for i=1:c.num_mfcc_filt,
```

```
            c.mel_filter(i,:) = ...
                (f > freqs(i) & f <= freqs(i+1)).* ...
                h(i).*(f-freqs(i))/(freqs(i+1)-freqs(i)) + ...
                (f > freqs(i+1) & f < freqs(i+2)).* ...
                h(i).*(freqs(i+2)-f)/(freqs(i+2)-freqs(i+1));
        end

        c.DCT = 1/sqrt(c.num_mfcc_filt/2) * ...
            cos((0:c.num_mfcc_coeffs-1)'*(0.5:c.num_mfcc_filt)*pi/c.num_mfcc_filt);
        c.DCT(1,:) = c.DCT(1,:)*sqrt(2)/2;
        c.w = 0.5*(1-cos(2*pi*(0:c.seg_size-1)/(c.seg_size-1)))'; %Hanning window

%% START FEATURE EXTRACTION CODE
       %compute P
            wav_temp = wav_temp * (10^(96/20));

            %considers the case of very soft audio audio parts
            temp11=abs(wav_temp);
            temp12=find(temp11<=(10^(3/20)));
            temp15=10*(10^(3/20))*randn(length(temp12),1);
            wav_temp(temp12)=temp15; %try to amplify and replace very soft parts

            num_segments    = floor((length(wav_temp)-c.seg_size)/c.hop_size)+1;
            P               = zeros(c.seg_size/2+1,num_segments); %% allocate memory
            X1              = zeros(c.seg_size/2+1,num_segments); %% allocate memory

            for i_p = 1:num_segments,
                idx = (1:c.seg_size)+(i_p-1)*c.hop_size;
                X = abs(fft(wav_temp(idx).*c.w));
                P(:,i_p) = 2*(X(1:c.seg_size/2+1).^2)/c.seg_size; %power per bin per frame
                X1(:,i_p)= 2*(X(1:c.seg_size/2+1))/c.seg_size;
            end

        clear wav_temp wav_bank temp11 temp12 temp15
        % compute M dB
            num_segments = size(P,2);
            M = zeros(c.num_mfcc_filt,num_segments);

            for i_m = 1:num_segments,
                M(:,i_m) = c.mel_filter*P(:,i_m);
            end
            M(M<1)=1; M = 10*log10(M);
            temp_zero = any(M);
            temp_zero = find(temp_zero==0);
            M=M(:,any(M)); %remove zero columns


        mfcc = c.DCT * M;

        %compute spectral centroid
            vsc     = ([0:size(P,1)-1]*P)./sum(P,1);

            % avoid NaN for silence frames
            vsc(sum(P,1) == 0) = 0;

            % convert from index to Hz
            vsc     = vsc / size(P,1) * c.fs/2;

            %normalize data
            %vsc = vsc/c.fs/2;


        %compute spectral spread
            % get spectral centroid as index

            % allocate memory
            vss     = zeros(size(vsc));

            % compute spread
            for n = 1:size(P,2)
                vss(n)  = (([0:size(P,1)-1]-vsc(n)).^2*P(:,n))./sum(P(:,n));
            end
            vss     = sqrt(vss);

            % convert from index to Hz
            vss     = vss / size(P,1) * c.fs/2;


        %compute spectral kurtosis and spectral skewness
            X1_temp = X1;
            % compute mean and standard deviation
            mu_x    = mean(abs(X1_temp), 1);
            std_x   = std(abs(X1_temp), 1);

            % remove mean
            X1_temp       = X1_temp - repmat(mu_x, size(X1_temp,1), 1);

            % compute kurtosis
            vsk     = sum ((X1_temp.^4)./(repmat(std_x, size(X1_temp,1), 1).^4*size(X1_temp,1)));
            vsk     = vsk-3;

            % compute skewness
            vssk    = sum ((X1_temp.^3)./(repmat(std_x, size(X1_temp,1), 1).^3*size(X1_temp,1)));

        clear X1_temp


        %compute spectral flatness
            XLog    = log(X1+1e-20);
            vtf     = exp(mean(XLog,1)) ./ mean(X1,1);


        %compute spectral flux
```

```
            % difference spectrum (set first diff to zero)
            afDeltaX    = diff([X1(:,1), X1],1,2);

            % spectral flux
            vsf         = sqrt(sum(afDeltaX.^2))/size(X1,1);


        %compute spectral rolloff
            % allocate memory
            vsr     = zeros(1,size(P,2));

            %compute rolloff
            afSum   = sum(P,1);
            for n = 1:length(vsr)
                vsr(n)   = find(cumsum(P(:,n)) >= c.kappa*afSum(n), 1);
            end

            % convert from index to Hz
            vsr     = vsr / size(P,1) * c.fs/2;

             %normalize data
            vsr = vsr/c.fs/2;


        %compute spectral brightness
            vbr     = sum(P(36:257,:))./sum(P,1);

            % avoid NaN for silence frames
            vbr(sum(P,1) == 0) = 0;


        %compute spectral entropy
            mn = P;
            % Negative data is trimmed:
            mn(mn<0) = 0;
             % Data is normalized such that the sum is equal to 1.
            mn = mn./repmat(sum(mn)+repmat(1e-12,...
                                [1 size(mn,2) size(mn,3) size(mn,4)]),...
                            [size(mn,1) 1 1 1]);

            % Actual computation of entropy
            vse = -sum(mn.*log(mn + 1e-12))./log(size(mn,1));


        aggre              = [vsc; vss; vsk; vssk; vtf; vsf; vsr; vbr; vse];
        aggre(:,temp_zero) = [];

        aggre           = aggre(:,any(aggre)); %remove zero columns
        aggre           = [mfcc;aggre];
        feat_m          = mean(aggre,2)';
        temp            = cov(aggre');
        feat_co         = temp;
        feat_ico        = inv(feat_co);
        feat_max_ico    = max(feat_ico(:));

end

function [feat_m,feat_co,feat_ico,feat_max_ico] = ExtractSBF(wav,Hd)
%%
%%  extracts sub-band flux features
%%

%% CONSTANTS FOR FEATURE EXTRACTION
        bands               = 10;  % number of sub-bands
        c.fs                = 22050;

        c.num_flux_filt     = bands;  % number of Mel frequency bands for flux
        c.num_flux_coeffs   = bands;  % number of flux coeffs

        c.seg_size          = 512; % 23ms if c.fs == 22050Hz
        c.hop_size          = 512;
        c.norm_freq         = linspace(0,c.fs,c.seg_size)/c.fs;

        c.w = 0.5*(1-cos(2*pi*(0:c.seg_size-1)/(c.seg_size-1)))'; %Hanning window
        c.DCT_flux = 1/sqrt(c.num_flux_filt/2) * ...
            cos((0:c.num_flux_coeffs-1)'*(0.5:c.num_flux_filt)*pi/c.num_flux_filt);
        c.DCT_flux(1,:) = c.DCT_flux(1,:)*sqrt(2)/2;

        %% START FEATURE EXTRACTION CODE

        if 1, %compute P
            wav = wav * (10^(96/20));

            %considers the case of very soft audio audio parts
            temp11=abs(wav);
            temp12=find(temp11<=(10^(3/20)));
            temp15=10*(10^(3/20))*randn(length(temp12),1);
            wav(temp12)=temp15; %try to amplify and replace very soft parts

            % filter sound file using filter bank
            wav_bank = zeros(length(wav),c.num_flux_filt);
            for i_m = 1:c.num_flux_filt
                wav_bank(:,i_m)=filter(Hd(i_m),wav);
            end

            num_segments    = floor((length(wav)-c.seg_size)/c.hop_size)+1;

            P               = zeros(c.seg_size/2+1,num_segments); %% allocate memory
            S_bank = zeros(c.num_flux_filt,c.seg_size/2+1,num_segments);

            for i_p = 1:num_segments,
                idx = (1:c.seg_size)+(i_p-1)*c.hop_size;
```

```
                X = abs(fft(wav(idx).*c.w));
                P(:,i_p)=2*(X(1:c.seg_size/2+1).^2)/c.seg_size; %power per bin per frame

                for i_b = 1:c.num_flux_filt
                    X = abs(fft(wav_bank(idx,i_b).*c.w));
                    S_bank(i_b,:,i_p)=2*(X(1:c.seg_size/2+1))/c.seg_size;
                end
            end
        end
        clear wav wav_bank temp11 temp12 temp15


        if 1, %compute sub-band flux coefficients
            temp_norm = zeros(c.num_flux_filt,num_segments);
            for i_b = 1:c.num_flux_filt
                temp        = S_bank(i_b,:,:);
                temp        = squeeze(temp);
               % difference spectrum (set first diff to zero)
                afDeltaX    = diff([temp(:,1), temp],1,2);
                % flux
                vsf         = sqrt(sum(afDeltaX.^2))/size(temp,1);
                temp_norm(i_b,:) = vsf;

            end
        end
        FLX = c.DCT_flux*temp_norm;
        temp    = FLX(:,any(FLX)); %remove zero columns
        feat_m      = mean(temp,2)';
        feat_co     = cov(temp');
        feat_ico    = inv(feat_co);
        feat_max_ico  = max(feat_ico(:));


end


function [feat_m,feat_co,feat_ico,feat_max_ico] = ExtractSC(wav)
%%
%% extract spectral contrast features
%%

%% CONSTANTS FOR FEATURE EXTRACTION
        coeffs = 20;
        c.fs                = 22050;
        c.alpha             = 0.2; % defines number of samples for SC
        c.num_sc_coeffs     = coeffs;
        c.num_filt          = coeffs;  % number of Mel frequency bands for SC

        c.num_flux_filt     = 10;  % number of Mel frequency bands for flux
        c.num_flux_coeffs   = 10;  % number of flux coeffs

        c.seg_size          = 512; % 23ms if c.fs == 22050Hz
        c.hop_size          = 512;
        c.norm_freq         = linspace(0,c.fs,c.seg_size)/c.fs;

        f = linspace(0,c.fs/2,c.seg_size/2+1); %% frequency bins of P
        mel = log(1+f/700)*1127.01048; %orig mel scale
        mel_idx = linspace(0,mel(end),c.num_filt+2);

        f_idx = zeros(c.num_filt+2,1);
        for i=1:c.num_filt+2,
            [~, f_idx(i)] = min(abs(mel - mel_idx(i)));
        end
        freqs = f(f_idx);

        % height of triangles
        h = 2./(freqs(3:c.num_filt+2)-freqs(1:c.num_filt));
        %h = ones(1,c.num_filt);

        c.mel_filter_sc = zeros(c.num_filt,c.seg_size/2+1);
        for i=1:c.num_filt,
            c.mel_filter_sc(i,:) = ...
                (f > freqs(i) & f <= freqs(i+1)).* ...
                h(i).*(f-freqs(i))/(freqs(i+1)-freqs(i)) + ...
                (f > freqs(i+1) & f < freqs(i+2)).* ...
                h(i).*(freqs(i+2)-f)/(freqs(i+2)-freqs(i+1));
            c.mel_length(i) = length(find(c.mel_filter_sc(i,:)));
        end

        c.w = 0.5*(1-cos(2*pi*(0:c.seg_size-1)/(c.seg_size-1)))'; %Hanning window
        c.DCT_sc = 1/sqrt(c.num_filt/2) * ...
            cos((0:c.num_sc_coeffs-1)'*(0.5:c.num_filt)*pi/c.num_filt);
        c.DCT_sc(1,:) = c.DCT_sc(1,:)*sqrt(2)/2;


        %% START FEATURE EXTRACTION CODE
        wav_temp = wav;

        if 1, %compute P
            wav_temp = wav_temp * (10^(96/20));

            %considers the case of very soft audio audio parts
            temp11=abs(wav_temp);
            temp12=find(temp11<=(10^(3/20)));
            temp15=10*(10^(3/20))*randn(length(temp12),1);
            wav_temp(temp12)=temp15; %try to amplify and replace very soft parts

            num_segments    = floor((length(wav_temp)-c.seg_size)/c.hop_size)+1;
            P               = zeros(c.seg_size/2+1,num_segments); %% allocate memory

            for i_p = 1:num_segments,
                idx = (1:c.seg_size)+(i_p-1)*c.hop_size;
                X = abs(fft(wav_temp(idx).*c.w));
```

```
                P(:,i_p) = 2*(X(1:c.seg_size/2+1).^2)/c.seg_size; %power per bin per frame
            end

        end
        clear wav_temp wav_bank temp11 temp12 temp15

        %compute spectral contrast
        sc = zeros(c.num_filt,num_segments);

        for i_m = 1:num_segments,
            temp = c.mel_filter_sc.*repmat(P(:,i_m),1,c.num_filt)';
            for i_n = 1: c.num_filt
                [~,~,temp1] = find(temp(i_n,:));
                temp1 = sort(temp1,'descend');
                id_pk = 1:ceil(c.alpha*c.mel_length(i_n));
                id_vl = c.mel_length(i_n)-ceil(c.alpha*c.mel_length(i_n))+1:c.mel_length(i_n);
                pk = log(1/(ceil(c.alpha*c.mel_length(i_n)))*sum(temp1(id_pk)));
                vl = log(1/(ceil(c.alpha*c.mel_length(i_n)))*sum(temp1(id_vl)));
                sc(i_n,i_m) = pk - vl;
            end
        end
        SC = c.DCT_sc*sc;

        temp    = SC(:,any(SC)); %remove zero columns
        feat_m = mean(temp,2)';
        feat_co = cov(temp');
        feat_ico = inv(feat_co);
        feat_max_ico = max(feat_ico(:));


end


function ComputeDistance(in_dir,out_file)
%%
%% USAGE EXAMPLE
%%       ComputeDistance(in_dir,out_file)
%%       ComputeDistance('mypath/myOutputDirectory','mypath/myResults.txt')
%%
%% INPUT ARGUMENTS
%%
%% in_dir:   directory to which function "ExtractFeatures" was writing to.
%%           a log file will be created in this directory
%% out_file: whole distance matrix in the following format
%%
%%   <start file, exclude this line>
%%     <SUBMISSION_NAME>
%%     1   path\file1.wav
%%     2   path\file2.wav
%%     ...
%%     N  path\fileN.wav
%%     Q\R   1       2      ...      N
%%     1     0.000  10.234  ...   123.32
%%     2    10.234   0.000  ...    23.45
%%     .     ...     ...    0.000   ...
%%     N     4.1234  6.345  ...     0.0
%%   <end file, exclude this line>
%%
%%   delimiter: tabulator space, number format: float
%%
%% HARDCODED PARAMETERS

if nargin~=2,
    error('Number of input arguments is not 2. (try "help ComputeDistance")')
end

if in_dir(end)~='/' && in_dir(end)~='\',
    in_dir(end+1)='/';
end
in_file = [in_dir,'Audio_features.mat'];

%% TEST WRITE ACCESS TO OUTPUT FILE
fid = fopen(out_file,'w');
if fid==-1, error('cannot write to output file (distance matrix)'); end
fprintf(fid,'%s','testwrite'); fclose(fid);
delete(out_file);

%% TEST INPUT FILE
fid = fopen(in_file,'r');
if fid==-1, error('cannot find file with extracted features (path problem?)'); end
fclose(fid);
load(in_file)

%% START LOGFILE
logfile = [in_dir,'ComputeDistance-',data.submission_name,'-logfile.txt'];
fid = fopen(logfile,'a');
if fid==-1, error('can''t append logfile'); end
fclose(fid);

mydisp(logfile,datestr(now));
mydisp(logfile, '-> ComputeDistance called.')
mydisp(logfile,['   Input directory: ',in_dir])
mydisp(logfile,'         (writing log-file to input directory)')


try %% big try catch to catch every error, write it to logfile and exit
    t0 = cputime;

    mydisp(logfile,'start computing spectral distances ...')
    num_files=length(data.filenames);

    SpecDist    = ComputeSpectralDistance(in_dir);
```

```
    mydisp(logfile,'start normalizing spectral distances ...')
    SpecDist_norm      = Norm_dist_mat(SpecDist);
    clear SpecDist

    mydisp(logfile,'start computing SBF distances ...')
    SBFDist     = ComputeSBFDistance(in_dir);

    mydisp(logfile,'start normalizing SBF distances ...')
    SBFDist_norm    = Norm_dist_mat(SBFDist);
    clear SBFDist

    mydisp(logfile,'start computing SC distances ...')
    SCDist      = ComputeSCDistance(in_dir);

    mydisp(logfile,'start normalizing SC distances ...')
    SCDist_norm    = Norm_dist_mat(SCDist);
    clear SCDist

    mydisp(logfile,'combining distances')
    D = 0.5*SpecDist_norm + SCDist_norm + 0.2*SBFDist_norm;
    %save D_out D

    mydisp(logfile,'done. start writing output ...')

    fid = fopen(out_file,'w');
    fprintf(fid,'%s\r\n',data.submission_name);
    for i=1:num_files,
        fprintf(fid,'%d\t%s\r\n',i,data.filenames{i});
    end
    fprintf(fid,'%s','Q\R');
    fprintf(fid,'\t%d',1:num_files);
    fprintf(fid,'\r\n');
    for i=1:num_files,
        fprintf(fid,'%d',i);
        fprintf(fid,'\t%d',D(i,:));
        fprintf(fid,'\r\n');
    end
    fclose(fid);

    mydisp(logfile,'output file created.')
    tot_time    = cputime-t0;
    mydisp(logfile,['total CPU time [h] ',num2str(tot_time/60/60)])

    mydisp(logfile,'done. exiting ...')
    mydisp(logfile,datestr(now))

catch exception
    mydisp(logfile,'-- caught error!')
    mydisp(logfile,lasterr)
    mydisp(logfile,datestr(now))

end
end

function mydisp(logfile,str)
fid = fopen(logfile,'a'); disp(str); fprintf(fid,'%s\r\n',str); fclose(fid);
end

function SpecDist = ComputeSpectralDistance(in_dir)
%%
%% compute MFCC and spectral distance

%% HARDCODED PARAMETERS

data.submission_name = 'DM1';
%%

if in_dir(end)~='/' && in_dir(end)~='\',
    in_dir(end+1)='/';
end
in_file = [in_dir,'Audio_features.mat'];

%% TEST INPUT FILE
fid = fopen(in_file,'r');
if fid==-1, error('cannot find file with extracted features (path problem?)'); end
fclose(fid);
load(in_file)

num_files=length(data.filenames);
D_g1b_kl = zeros(num_files,num_files);

t0 = cputime;
t1 = cputime; %this is updated every iteration
num_computations = num_files*(num_files-1)/2;
num_computations_sofar = 0;

for i=1:num_files-1,
    t2 = cputime;
    if t2-t1>5, %% output current status only every 5 seconds (cputime)
        %tmp_avg_t_sofar = (t1-t0)/(num_computations_sofar-1);
        tmp_est_tot = (t1-t0)/(num_computations_sofar-1)*num_computations;
        tmp_est_rem = tmp_est_tot - (t1-t0);
        disp([num2str(num_computations_sofar),'/',num2str(num_computations), ...
            ' SpecDist est rem ',num2str(tmp_est_rem/60),'m, est tot ', ...
            num2str(tmp_est_tot/60),'m']);
        t1 = cputime;
    end
    for j=i+1:num_files,
        d_g1_computed = false;

        if all([data.feat.g1c.aggre_ico(i),data.feat.g1c.aggre_ico(j)]<10^10),
            tmp = squeeze(data.feat.aggre.m(i,:,:))-squeeze(data.feat.aggre.m(j,:,:));
            d_g1b = ... %% kl distance
```

```
                trace(squeeze(data.feat.aggre.co(i,:,:))*squeeze(data.feat.aggre.ico(j,:,:))) + ...
                trace(squeeze(data.feat.aggre.co(j,:,:))*squeeze(data.feat.aggre.ico(i,:,:))) + ...
                trace((squeeze(data.feat.aggre.ico(i,:,:))+squeeze(data.feat.aggre.ico(j,:,:)))*(tmp')*tmp);
            d_g1_computed = true;
        end

        if d_g1_computed,
            D_g1b_kl(i,j) = sqrt(log(1+d_g1b)); %transform answer to metric

        else %% not evaluated work around to deal with inv covariance problems FIX THIS

            D_g1b_kl(i,j) = 100;

        end
        num_computations_sofar = num_computations_sofar+1;
    end
end

SpecDist = D_g1b_kl + D_g1b_kl';

end


function SBFDist = ComputeSBFDistance(in_dir)
%%
%% compute SBF distance

%% HARDCODED PARAMETERS

data.submission_name = 'DM1';
%%
if in_dir(end)~='/' && in_dir(end)~='\',
    in_dir(end+1)='/';
end
in_file = [in_dir,'Audio_features.mat'];

%% TEST INPUT FILE
fid = fopen(in_file,'r');
if fid==-1, error('cannot find file with extracted features (path problem?)'); end
fclose(fid);
load(in_file)


num_files=length(data.filenames);


D_sbf_kl = zeros(num_files,num_files);


t0 = cputime;
t1 = cputime; %this is updated every iteration
num_computations = num_files*(num_files-1)/2;
num_computations_sofar = 0;

for i=1:num_files-1,
    t2 = cputime;
    if t2-t1>5, %% output current status only every 5 seconds (cputime)
        %tmp_avg_t_sofar = (t1-t0)/(num_computations_sofar-1);
        tmp_est_tot = (t1-t0)/(num_computations_sofar-1)*num_computations;
        tmp_est_rem = tmp_est_tot - (t1-t0);
        disp([num2str(num_computations_sofar),'/',num2str(num_computations), ...
            ' SBFDist est rem ',num2str(tmp_est_rem/60),'m, est tot ', ...
            num2str(tmp_est_tot/60),'m'])
        t1 = cputime;
    end
    for j=i+1:num_files,
        d_sbf_computed = false;

        if all([data.feat.g1c.flx_ico(i),data.feat.g1c.flx_ico(j)]<10^10),
            tmp = squeeze(data.feat.flux.m(i,:,:))-squeeze(data.feat.flux.m(j,:,:));
            d_sbf = ... %% kl distance
                trace(squeeze(data.feat.flux.co(i,:,:))*squeeze(data.feat.flux.ico(j,:,:))) + ...
                trace(squeeze(data.feat.flux.co(j,:,:))*squeeze(data.feat.flux.ico(i,:,:))) + ...
                trace((squeeze(data.feat.flux.ico(i,:,:))+squeeze(data.feat.flux.ico(j,:,:)))*(tmp')*tmp);
            d_sbf_computed = true;
        end


        if d_sbf_computed,
            D_sbf_kl(i,j) = sqrt(log(1+d_sbf)); %transform answer to metric


        else %% not evaluated work around to deal with inv covariance problems FIX THIS

            D_sbf_kl(i,j) = 100;


        end
        num_computations_sofar = num_computations_sofar+1;
    end
end

SBFDist = D_sbf_kl + D_sbf_kl';

end

function SCDist = ComputeSCDistance(in_dir)
%%
%% compute SC distance

%% HARDCODED PARAMETERS

data.submission_name = 'DM1';
```

```
%%
if in_dir(end)~='/' && in_dir(end)~='\',
    in_dir(end+1)='/';
end
in_file = [in_dir,'Audio_features.mat'];

%% TEST INPUT FILE
fid = fopen(in_file,'r');
if fid==-1, error('cannot find file with extracted features (path problem?)'); end
fclose(fid);
load(in_file)


num_files=length(data.filenames);


D_sc_kl = zeros(num_files,num_files);


t0 = cputime;
t1 = cputime; %this is updated every iteration
num_computations = num_files*(num_files-1)/2;
num_computations_sofar = 0;

for i=1:num_files-1,
    t2 = cputime;
    if t2-t1>5, %% output current status only every 5 seconds (cputime)
        %tmp_avg_t_sofar = (t1-t0)/(num_computations_sofar-1);
        tmp_est_tot = (t1-t0)/(num_computations_sofar-1)*num_computations;
        tmp_est_rem = tmp_est_tot - (t1-t0);
        disp([num2str(num_computations_sofar),'/',num2str(num_computations), ...
            ' CompDist est rem ',num2str(tmp_est_rem/60),'m, est tot ', ...
            num2str(tmp_est_tot/60),'m'])
        t1 = cputime;
    end
    for j=i+1:num_files,
        d_sc_computed = false;


        if all([data.feat.g1c.sc_ico(i),data.feat.g1c.sc_ico(j)]<10^10),
            tmp = squeeze(data.feat.sc.m(i,:,:))-squeeze(data.feat.sc.m(j,:,:));
            d_sc = ... %% kl distance
                trace(squeeze(data.feat.sc.co(i,:,:))*squeeze(data.feat.sc.ico(j,:,:))) + ...
                trace(squeeze(data.feat.sc.co(j,:,:))*squeeze(data.feat.sc.ico(i,:,:))) + ...
                trace((squeeze(data.feat.sc.ico(i,:,:))+squeeze(data.feat.sc.ico(j,:,:)))*(tmp')*tmp);
            d_sc_computed = true;
        end


        if d_sc_computed,
            D_sc_kl(i,j) = sqrt(log(1+d_sc)); %transform answer to metric

        else %% not evaluated work around to deal with inv covariance problems FIX THIS

            D_sc_kl(i,j) = 100;

        end
        num_computations_sofar = num_computations_sofar+1;
    end
end

SCDist = D_sc_kl + D_sc_kl';

end

function D_norm = Norm_dist_mat(D)

num_files = length(D);
D_norm = zeros(num_files,num_files);
t1 = cputime;

for i=1:num_files %distance space normalizatiion wrt row,col
    t2 = cputime;
    for j=1:num_files
        temp1=[D(i,:) D(:,j)'];
        D_norm(i,j)=(D(i,j)-mean(temp1))/std(temp1);

    end
    if rem(i,20)==0,
        time.avg_t_sofar = (t2-t1)/(i-1);
            time.est_tot = (t2-t1)/(i-1)*num_files;
            time.est_rem = time.est_tot - (t2-t1);
            disp([num2str(i),'/',num2str(num_files), ...
                ' Normalization est rem ',num2str(time.est_rem/60),'m, est tot ', ...
                num2str(time.est_tot/60),'m'])
    end
end

D_norm = D_norm+10;
D_norm(find(eye(num_files)))=zeros(1,num_files);

end

function [X,PA]=FastMap(in_dir,k)
%%
%% USAGE EXAMPLE
%%       FastMap(in_dir,k)
%%       ComputeDistance('mypath/myOutputDirectory',10)
%%
%% INPUT ARGUMENTS
%%
%% in_dir:   directory to which function "AudioFeatures.mat" is located.
%% k: target Euclidean space dimensionality
```

```
%%
%%
%% HARDCODED PARAMETERS
%in_dir='C:\Users\Franz\Documents\MATLAB\MIREX2011\AMS update\output';
if in_dir(end)~='/' && in_dir(end)~='\',
    in_dir(end+1)='/';
end
in_file = [in_dir,'Audio_features.mat'];

%% TEST INPUT FILE
fid = fopen(in_file,'r');
if fid==-1, error('cannot find file with extracted features (path problem?)'); end
fclose(fid);
load(in_file)

n=length(data.feat.aggre.m);
%k=5;
X=zeros(n,k);
PA=zeros(2,k);
col=0;
%%
while k>0
    k
    col=col+1;

    if col==1
        %select pivot objects
        pivotB = randi(n,n,1);
        pivotB = pivotB(1);  %select random pivot B

        %for m = 1:2
        %select farthest object from pivot B as pivot A

        i=1;
        dis=zeros(1,n);
        while i<=n
            if i~=pivotB
                %tmp = data(i).timbre'-data(pivotB).timbre';
                tmp = squeeze(data.feat.aggre.m(i,:,:))-squeeze(data.feat.aggre.m(pivotB,:,:));
                dis(i) = ... %% kl distance
                    trace(squeeze(data.feat.aggre.co(i,:,:))*squeeze(data.feat.aggre.ico(pivotB,:,:))) + ...
                    trace(squeeze(data.feat.aggre.co(pivotB,:,:))*squeeze(data.feat.aggre.ico(i,:,:))) + ...
                    trace((squeeze(data.feat.aggre.ico(i,:,:))+squeeze(data.feat.aggre.ico(pivotB,:,:)))*(tmp')*tmp);
                i=i+1;
            else
                i=i+1;
            end
        end
        [~,I]=sort(dis);
        pivotA=I(ceil(n/2));
        %[~,pivotA]=max(dis);


        %select farthest object from pivot A as pivot B
        i=1;
        dis=zeros(1,n);
        while i<=n
            if i~=pivotA
                tmp = squeeze(data.feat.aggre.m(i,:,:))-squeeze(data.feat.aggre.m(pivotA,:,:));
                dis(i) = ... %% kl distance
                    trace(squeeze(data.feat.aggre.co(i,:,:))*squeeze(data.feat.aggre.ico(pivotA,:,:))) + ...
                    trace(squeeze(data.feat.aggre.co(pivotA,:,:))*squeeze(data.feat.aggre.ico(i,:,:))) + ...
                    trace((squeeze(data.feat.aggre.ico(i,:,:))+squeeze(data.feat.aggre.ico(pivotA,:,:)))*(tmp')*tmp);
                i=i+1;
            else
                i=i+1;
            end
        end
        [~,I]=sort(dis);
        pivotB=I(ceil(n/2));
        %[~,pivotB]=max(dis);
        %end

        %record IDs of the pivot objects
        PA(1,col)=pivotA;   PA(2,col)=pivotB;
        tmp = squeeze(data.feat.aggre.m(pivotB,:,:))-squeeze(data.feat.aggre.m(pivotA,:,:));
        dAB = ... %% kl distance
            trace(squeeze(data.feat.aggre.co(pivotB,:,:))*squeeze(data.feat.aggre.ico(pivotA,:,:))) + ...
            trace(squeeze(data.feat.aggre.co(pivotA,:,:))*squeeze(data.feat.aggre.ico(pivotB,:,:))) + ...
            trace((squeeze(data.feat.aggre.ico(pivotB,:,:))+squeeze(data.feat.aggre.ico(pivotA,:,:)))*(tmp')*tmp);
        dAB = sqrt(log(1+dAB));

        %given pivotA, pivotB, i
        %project objects on line (Oa,Ob) for each object Oi
        for i=1:n
            if i==pivotA
                dAi=0;

            else
                tmp = squeeze(data.feat.aggre.m(i,:,:))-squeeze(data.feat.aggre.m(pivotA,:,:));
                dAi = ... %% kl distance
                    trace(squeeze(data.feat.aggre.co(i,:,:))*squeeze(data.feat.aggre.ico(pivotA,:,:))) + ...
                    trace(squeeze(data.feat.aggre.co(pivotA,:,:))*squeeze(data.feat.aggre.ico(i,:,:))) + ...
                    trace((squeeze(data.feat.aggre.ico(i,:,:))+squeeze(data.feat.aggre.ico(pivotA,:,:)))*(tmp')*tmp);
                dAi = sqrt(log(1+dAi));
            end

            if i==pivotB
                dBi=0;

            else
                tmp =  squeeze(data.feat.aggre.m(i,:,:))-squeeze(data.feat.aggre.m(pivotB,:,:));
                dBi = ... %% kl distance
```

```
                trace(squeeze(data.feat.aggre.co(i,:,:))*squeeze(data.feat.aggre.ico(pivotB,:,:))) + ...
                trace(squeeze(data.feat.aggre.co(pivotB,:,:))*squeeze(data.feat.aggre.ico(i,:,:))) + ...
                trace((squeeze(data.feat.aggre.ico(i,:,:))+squeeze(data.feat.aggre.ico(pivotB,:,:)))*(tmp')*tmp);
            dBi = sqrt(log(1+dBi));
        end

        %temp(i)= (dAi^2+dAB^2-dBi^2)/(2*dAB);
        X(i,col)= (dAi^2+dAB^2-dBi^2)/(2*dAB);  %update projection matrix
    end

    %X(:,col)=temp';
    k=k-1;


else
    %select pivot objects
    pivotB = randi(n,n,1);
    pivotB = pivotB(1);  %select random pivot B

    %for m=1:2
    %select farthest object from pivot B as pivot A
    dist_pivot_sq=zeros(1,n);

    for i= 1:n
        counter=col;

        if i==pivotB
            dBi=0;

        else
            tmp = squeeze(data.feat.aggre.m(i,:,:))-squeeze(data.feat.aggre.m(pivotB,:,:));
            dBi = ... %% kl distance
                trace(squeeze(data.feat.aggre.co(i,:,:))*squeeze(data.feat.aggre.ico(pivotB,:,:))) + ...
                trace(squeeze(data.feat.aggre.co(pivotB,:,:))*squeeze(data.feat.aggre.ico(i,:,:))) + ...
                trace((squeeze(data.feat.aggre.ico(i,:,:))+squeeze(data.feat.aggre.ico(pivotB,:,:)))*(tmp')*tmp);
            dBi = sqrt(log(1+dBi));
        end

        temp = dBi^2;
        while counter>1
            temp = temp -(X(pivotB,counter-1)-X(i,counter-1))^2;
            counter = counter - 1;
        end
        dist_pivot_sq(i) = temp;

    end
    [~,I]=sort(dist_pivot_sq);
    pivotA=I(ceil(n/2));
    %[~,pivotA]=max(dist_pivot_sq);

    %select farthest object from pivot A as pivot B
    for i= 1:n
        counter=col;

        if i==pivotA
            dAi=0;

        else
            tmp = squeeze(data.feat.aggre.m(i,:,:))-squeeze(data.feat.aggre.m(pivotA,:,:));
            dAi = ... %% kl distance
                trace(squeeze(data.feat.aggre.co(i,:,:))*squeeze(data.feat.aggre.ico(pivotA,:,:))) + ...
                trace(squeeze(data.feat.aggre.co(pivotA,:,:))*squeeze(data.feat.aggre.ico(i,:,:))) + ...
                trace((squeeze(data.feat.aggre.ico(i,:,:))+squeeze(data.feat.aggre.ico(pivotA,:,:)))*(tmp')*tmp);
            dAi = sqrt(log(1+dAi));
        end

        temp = dAi^2;
        while counter>1
            temp = temp -(X(pivotA,counter-1)-X(i,counter-1))^2;
            counter = counter - 1;
        end
        dist_pivot_sq(i)=temp;
    end
    [~,I]=sort(dist_pivot_sq);
    pivotB=I(ceil(n/2));
    %[~,pivotB]=max(dist_pivot_sq);
    %end

    %record IDs of the pivot objects
    PA(1,col)=pivotA;   PA(2,col)=pivotB;

    %remember projected distence between i, A and B
    distAi = dist_pivot_sq;
    distAB = distAi(pivotB);
    for i= 1:n
        counter=col;

        if i==pivotB
            dBi=0;

        else
            dBi = ... %% kl distance
               trace(squeeze(data.feat.aggre.co(i,:,:))*squeeze(data.feat.aggre.ico(pivotB,:,:))) + ...
                trace(squeeze(data.feat.aggre.co(pivotB,:,:))*squeeze(data.feat.aggre.ico(i,:,:))) + ...
                trace((squeeze(data.feat.aggre.ico(i,:,:))+squeeze(data.feat.aggre.ico(pivotB,:,:)))*(tmp')*tmp);
            dBi = sqrt(log(1+dBi));
        end

        temp = dBi^2;
        while counter>1
            temp =temp - (X(pivotB,counter-1)-X(i,counter-1))^2;
            counter = counter - 1;
        end
```

```
            dist_pivot_sq(i)=temp;

        end
        distBi = dist_pivot_sq;

        %project objects on line (Oa,Ob) for each object Oi
        for i=1:n
            X(i,col)=(distAi(i)+distAB-distBi(i))/...
                (2*sqrt(distAB));
            %temp(i)=(distAi(i)+distAB-distBi(i))/...
            %    (2*sqrt(distAB));
        end

        %update projection matrix
        %X(:,col)=temp';
        k=k-1;
    end

end
end
```

Appendix 3 DB-S Dataset Song List

Format: genre\artist – song title

1. acoustic\Beatles - Anna (Go to Him).wav
2. acoustic\Beatles - Ask Me Why.wav
3. acoustic\Beatles - Baby It's You.wav
4. acoustic\Beatles - Boys.wav
5. acoustic\Beatles - Chains.wav
6. acoustic\Beatles - I Saw Her Standing There.wav
7. acoustic\Beatles - Love Me Do.wav
8. acoustic\Beatles - Misery.wav
9. acoustic\Beatles - P.S. I Love You.wav
10. acoustic\Beatles - Please Please Me.wav
11. acoustic\Eraserheads - Easy ka lang.wav
12. acoustic\Eraserheads - Ganjazz.wav
13. acoustic\Eraserheads - Honky_Toinks_Granny.wav
14. acoustic\Eraserheads - Ligaya.wav
15. acoustic\Eraserheads - Maling akala.wav
16. acoustic\Eraserheads - Pare ko.wav
17. acoustic\Eraserheads - Shake yer head.wav
18. acoustic\Eraserheads - Shirley.wav
19. acoustic\Eraserheads - Tindahan ni Aling Nena.wav
20. acoustic\Eraserheads - Toyang.wav
21. acoustic\John Mayer - Born And Raised.wav
22. acoustic\John Mayer - If I Ever Get Around To Living.wav
23. acoustic\John Mayer - Love Is A Verb.wav
24. acoustic\John Mayer - Queen Of California.wav
25. acoustic\John Mayer - Shadow Days.wav
26. acoustic\John Mayer - Something Like Olivia.wav
27. acoustic\John Mayer - Speak For Me.wav
28. acoustic\John Mayer - The Age Of Worry.wav
29. acoustic\John Mayer - Walt Grace's Submarine Test January 1967.wav
30. acoustic\John Mayer - Whiskey Whiskey Whiskey.wav
31. acoustic\Sabrina - Airplanes.wav
32. acoustic\Sabrina - Baby.wav
33. acoustic\Sabrina - Billionaire.wav
34. acoustic\Sabrina - California Gurls.wav
35. acoustic\Sabrina - Gotten.wav
36. acoustic\Sabrina - Hey Soul Sister.wav
37. acoustic\Sabrina - Love The Way You Lie.wav
38. acoustic\Sabrina - Need You Now.wav
39. acoustic\Sabrina - Never Say Never.wav
40. acoustic\Sabrina - OMG.wav
41. acoustic\Taylor Swift-a_place_in_this_world.wav
42. acoustic\Taylor Swift-cold_as_you.wav
43. acoustic\Taylor Swift-marys_song_(oh_my_my_my).wav
44. acoustic\Taylor Swift-picture_to_burn.wav
45. acoustic\Taylor Swift-shouldve_said_no.wav
46. acoustic\Taylor Swift-stay_beautiful.wav
47. acoustic\Taylor Swift-teardrops_on_my_guitar.wav
48. acoustic\Taylor Swift-the_outside.wav
49. acoustic\Taylor Swift-tied_together_with_a_smile.wav
50. acoustic\Taylor Swift-tim_mcgraw.wav
51. classical\Bach - Air on a G string.wav
52. classical\Bach - Bouree in E Minor.wav
53. classical\Bach - Cello Suite 1.wav
54. classical\Bach - Jesu, Joy Of Man's Desiring.wav
55. classical\Bach - Minuet In G Major.wav
56. classical\Bach - Minuet.wav
57. classical\Bach - Prelude In C.wav
58. classical\Bach - Toccata and Fugue in D Minor.wav
59. classical\Bach- Violoncello Minuet.wav
60. classical\Beethoven - 5th symphony.wav
61. classical\Beethoven - 6th Symphony.wav
62. classical\Beethoven - 9th Ode to Joy.wav
63. classical\Beethoven - 9th Symphony.wav
64. classical\Beethoven - Fur Elise.wav
65. classical\Beethoven - Minuet in G.wav
66. classical\Beethoven - Moonlight Sonata.wav
67. classical\Beethoven - Pathetique.wav
68. classical\Beethoven - Piano Concerto no5.wav
69. classical\Chopin - Nocturne No 2.wav
70. classical\Clarke - Trumpet Voluntary.wav
71. classical\Debussy - Arabesque.wav
72. classical\Debussy - Claire de Lune.wav
73. classical\Elgar - Pomp and Circumstance March.wav
74. classical\Fucik - Entry Of The Gladiators.wav
75. classical\Handel - Arrival of the Queen of Sheba.wav
76. classical\Handel - Joy To The World.wav
77. classical\Handel - Suite no 2.wav
78. classical\Handel - The Harmonious Blacksmith.wav
79. classical\Handel - The Messiah-Hallelujah Chorus.wav
80. classical\Handel - Water Music.wav
81. classical\Mozart - Ah! Vous Dirai - Je, Maman.wav
82. classical\Mozart - Ave Verum Corpus.wav
83. classical\Mozart - Eine Kleine Nachtmusik.wav
84. classical\Mozart - Overture to The Marriage of Figaro.wav
85. classical\Mozart - Piano Concerto no20.wav
86. classical\Mozart - Piano Concerto no21.wav
87. classical\Mozart - Piano Sonata no11.wav
88. classical\Mozart - Piano Sonata no16.wav
89. classical\Mozart - Queen Of The Night.wav
90. classical\Mozart - Requiem.wav
91. classical\Pachelbel - Canon in D.wav

92. classical\Paganini - 24th Caprice.wav
93. classical\Paganini - 5th Caprice.wav
94. classical\Tchaikovsky - Romeo and Juliet.wav
95. classical\Tchaikovsky - Swan Lake.wav
96. classical\Vivaldi - Andante.wav
97. classical\Vivaldi - Concerto for Guitar.wav
98. classical\Vivaldi - Gloria In Excelsis Deo.wav
99. classical\Vivaldi - Spring.wav
100. classical\Vivaldi - Winter.wav
101. electronica\Bent - Magic Love (Ashley Beedle's Black Magic Remix Edit).wav
102. electronica\C.J. Bolland - Sugar Is Sweeter (Armand Van Helden's Drum n' Bass Mix).wav
103. electronica\Chicane - Saltwater.wav
104. electronica\Danny J Lewis - Spend The Night (H-Man Mix).wav
105. electronica\Darude - Sandstorm.wav
106. electronica\Delerium feat. Sarah Mclachlan - Silence (DJ Tiesto's In Search Of Sunrise Remix.wav
107. electronica\Double 99 - Ripgroove.wav
108. electronica\Example - Watch The Sun Come Up (Live Studio Version).wav
109. electronica\FC Kahuna - Hayling.wav
110. electronica\Faithless - God Is A DJ (Monster Mix).wav
111. electronica\Finley Quaye & William Orbit - Dice.wav
112. electronica\George Morel - Let's Groove.wav
113. electronica\Goldie - Inner City Life.wav
114. electronica\Groove Armada - At The River.wav
115. electronica\Hardrive - Deep Inside.wav
116. electronica\I Monster - Daydream In Blue.wav
117. electronica\Indo - RU Sleeping (Bump & Flex Remix).wav
118. electronica\Jakatta - American Dream (Afterlife Remix).wav
119. electronica\Jose Gonzalez - Heartbeats.wav
120. electronica\Lemon Jelly - Nice Weather For Ducks.wav
121. electronica\Lustral - Everytime (Nalin & Kane Remix).wav
122. electronica\MD X-press - God Made Me Phunky.wav
123. electronica\Mylo - In My Arms.wav
124. electronica\Nalin & Kane - Beachball (Extended Vocal Mix).wav
125. electronica\Nu-Birth - Anytime.wav
126. electronica\Paul Van Dyk - For An Angel (PvD 09 Remix).wav
127. electronica\Robert Miles - Children.wav
128. electronica\Robin S. - Show Me Love.wav
129. electronica\Roger Sanchez - Another Chance (Afterlife Mix).wav
130. electronica\Roy Davies Jr. Ft. Peven Everett - Gabriel (Live Garage Mix).wav
131. electronica\Royksopp - Poor Leno.wav
132. electronica\Rui Da Silva Ft. Cassandra - Touch Me.wav
133. electronica\Scott Garcia Ft. MC Styles - It's A London Thing.wav
134. electronica\Sneaker Pimps - Spin Spin Sugar (Armand's Dark Garage Mix).wav
135. electronica\Solu Music Ft. KimBlee - Fade.wav
136. electronica\Somore Ft. Damon Trueitt - I Refuse (What You Want)(Industry Standard Remix).wav
137. electronica\System F - Out Of The Blue.wav
138. electronica\The Avalanches - Since I Left You.wav
139. electronica\The Beloved - Sweet Harmony.wav
140. electronica\The Cinematic Orchestra - To Build A Home.wav
141. electronica\The Dreem Team - The Theme.wav
142. electronica\The Nightcrawlers - Push The Feeling On (Dub Of Doom).wav
143. electronica\The Sabres of Paradise - Smokebelch II (Beatless Mix).wav
144. electronica\Tina Moore - Never Gonna Let You Go (Kelly G. Bump-N-Go Mix).wav
145. electronica\Tori Amos - Professional Widow (Armand's Star Trunk Funk Mix).wav
146. electronica\Wookie Ft. Lain - Battle.wav
147. electronica\Wretch 32 Ft. Josh Kumra - Don't Go.wav
148. electronica\X-Press 2 - Lazy.wav
149. electronica\Y-Tribe - Enough Is Enough.wav
150. electronica\Yasmin - Finish Line.wav
151. hardrock\ACDC - Can I Sit Next To You Girl.wav
152. hardrock\ACDC - Gone Shootin'.wav
153. hardrock\ACDC - Heatseeker.wav
154. hardrock\ACDC - Highway to Hell.wav
155. hardrock\ACDC - If You Want Blood (You've Got It).wav
156. hardrock\ACDC - It's A Long Way To The Top (If You Wanna Rock 'n' Roll).wav
157. hardrock\ACDC - Jailbreak.wav
158. hardrock\ACDC - Let There Be Rock.wav
159. hardrock\ACDC - School Days.wav
160. hardrock\ACDC - Whole Lotta Rosie.wav
161. hardrock\GunsnRoses - Chinese Democracy.wav
162. hardrock\GunsnRoses - Paradise City.wav
163. hardrock\GunsnRoses - Since I Don't Have You.wav
164. hardrock\GunsnRoses - Sweet Child O' Mine.wav
165. hardrock\GunsnRoses - Welcome To The Jungle.wav
166. hardrock\GunsnRoses - Better.wav
167. hardrock\GunsnRoses - Don't Cry.wav
168. hardrock\GunsnRoses - Live And Let Die.wav
169. hardrock\GunsnRoses - You Could Be Mine.wav
170. hardrock\GunsnRoses - You're Crazy.wav
171. hardrock\Kiss - Calling Dr. Love.wav
172. hardrock\Kiss - I Stole Your Love.wav
173. hardrock\Kiss - I Was Made for Lovin' You.wav
174. hardrock\Kiss - Lick It Up.wav
175. hardrock\Kiss - Love Gun.wav
176. hardrock\Kiss - Parasite.wav
177. hardrock\Kiss - Rock Bottom.wav
178. hardrock\Kiss - Room Service.wav
179. hardrock\Kiss - Strutter.wav
180. hardrock\Kiss - Take Me.wav
181. hardrock\Metallica - Battery.wav
182. hardrock\Metallica - Enter Sandman.wav
183. hardrock\Metallica - Fade To Black.wav
184. hardrock\Metallica - Fuel.wav
185. hardrock\Metallica - Last Caress-Green Hell.wav
186. hardrock\Metallica - Metal Militia.wav
187. hardrock\Metallica - Sad But True.wav
188. hardrock\Metallica - Seek & Destroy.wav
189. hardrock\Metallica - The Small Hours.wav
190. hardrock\Metallica - Whiplash.wav
191. hardrock\RAGM - Bombtrack.wav
192. hardrock\RAGM - Bullet In The Head.wav
193. hardrock\RAGM - Clear The Lane.wav
194. hardrock\RAGM - Darkness Of Greed.wav

Appendix 3 DB-S Dataset Song List

195. hardrock\RAGM - Freedom.wav
196. hardrock\RAGM - Know Your Enemy.wav
197. hardrock\RAGM - Settle For Nothing.wav
198. hardrock\RAGM - Take The Power Back.wav
199. hardrock\RAGM - Township Rebellion.wav
200. hardrock\RAGM - Wake Up.wav
201. hiphop\Ante Up [Robbin Hoodz Theory] [.wav
202. hiphop\Back By Dope Demand.wav
203. hiphop\Born And Raised In Compton [Expli.wav
204. hiphop\C.R.E.A.M. [Explicit].wav
205. hiphop\California Love.wav
206. hiphop\Can I Kick It_ [Boilerhouse Mix].wav
207. hiphop\Dead Presidents II [Explicit].wav
208. hiphop\Deja Vu [Uptown Baby] [Explicit.wav
209. hiphop\Don't Scandalize Mine.wav
210. hiphop\Fu-Gee-La [Explicit].wav
211. hiphop\Get Ur Freak On [Explicit].wav
212. hiphop\Go See The Doctor.wav
213. hiphop\Got Ur Self A....wav
214. hiphop\Gravel Pit [Explicit].wav
215. hiphop\Grindin' [Explicit].wav
216. hiphop\Halftime [Explicit].wav
217. hiphop\Hard Knock Life [The Ghetto Ant.wav
218. hiphop\Harder Than You Think.wav
219. hiphop\Hip Hop [Explicit].wav
220. hiphop\How I Could Just Kill A Man [Expl.wav
221. hiphop\I Got It Made [Explicit].wav
222. hiphop\I Used To Love H.E.R [Explicit].wav
223. hiphop\Ice Cream [Explicit].wav
224. hiphop\Insane In The Brain [Explicit].wav
225. hiphop\It Takes Two.wav
226. hiphop\It's A Shame [Explicit].wav
227. hiphop\Jazz Thing [Video Mix].wav
228. hiphop\Jump Around.wav
229. hiphop\Lord Give Me A Sign.wav
230. hiphop\Make Room [Explicit].wav
231. hiphop\Ms. Jackson.wav
232. hiphop\My Philosophy.wav
233. hiphop\Nuthin' But A G Thang [Explicit].wav
234. hiphop\Peter Piper [Explicit].wav
235. hiphop\Put It On.wav
236. hiphop\Rock Dis Funky Joint [Explicit].wav
237. hiphop\Scenario [Explicit].wav
238. hiphop\Shook Ones Part II [Explicit].wav
239. hiphop\So Rotton.wav
240. hiphop\Sound Bway Bureill [Explicit].wav
241. hiphop\Step Into A World.wav
242. hiphop\Straight Out The Jungle.wav
243. hiphop\Treat 'em Right [Cribb Mix].wav
244. hiphop\Twinz (Deep Cover '98) [Explici.wav
245. hiphop\Vivrant Thing [Explicit].wav
246. hiphop\Walk This Way.wav
247. hiphop\Who Am I [What's My Name_] [Exp.wav
248. hiphop\Who Got Da Props.wav
249. hiphop\Witness [1 Hope] [Explicit].wav
250. hiphop\X [Explicit].wav

Appendix 4 MIREX 2013 AMS Result (DM1)

```
Retriever: DistanceMatrixRetriever - DM1 (dense)
Report time stamp: Fri Oct 25 12:37:53 CDT 2013

---
Neighbourhood clustering according to indexed metadatas:
        [% top 5 in same class, Random baseline, Normalised % top 5 in same class],
genre   0.7404285714285816      0.09988571428571427     0.7404285714285816
artist  0.43545714285713566     0.013046040816326553    0.43709999999999405
trackname       0.0     0.0     0.0
album   0.5108571428571429      0.05322473469387734     0.5110928571428572
        [% top 10 in same class, Random baseline, Normalised % top 10 in same class],
genre   0.6951142857142881      0.09988571428571427     0.6951142857142881
artist  0.35778571428570927     0.013046040816326553    0.37284058956916233
trackname       0.0     0.0     0.0
album   0.4431428571428582      0.05322473469387734     0.4495423469387792
        [% top 20 in same class, Random baseline, Normalised % top 20 in same class],
genre   0.6457928571428527      0.09988571428571427     0.6457928571428527
artist  0.2790714285714264      0.013046040816326553    0.33724049577259574
trackname       0.0     0.0     0.0
album   0.37297142857143056     0.05322473469387734     0.40908203251181535
        [% top 50 in same class, Random baseline, Normalised % top 50 in same class],
genre   0.5722314285714297      0.09988571428571427     0.5722314285714297
artist  0.18943142857142178     0.013046040816326553    0.3343164295410895
trackname       0.0     0.0     0.0
album   0.2884771428571403      0.05322473469387734     0.38254950900541246
---
Artist Filtered Genre Neighbourhood clustering:
                [% top 5 in same class, Normalised % top 5 in same class]
Filtered Genre  0.5350285714285754      0.5350285714285754
                [% top 10 in same class, Normalised % top 10 in same class]
Filtered Genre  0.5133428571428582      0.5133428571428582
                [% top 20 in same class, Normalised % top 20 in same class]
Filtered Genre  0.4893428571428565      0.4893428571428565
                [% top 50 in same class, Normalised % top 50 in same class]
Filtered Genre  0.45267428571428375     0.45267428571428375
---
Artist Filtered Genre Neighbourhood confusion:

Classes:
0:      METAL
1:      BLUES
2:      BAROQUE
3:      COUNTRY
4:      ROCKROLL
5:      JAZZ
6:      RAPHIPHOP
7:      EDANCE
8:      CLASSICAL
9:      ROMANTIC

At 5 results
q:        0       1       2       3       4       5       6       7       8        9
r:
0:      0.69486 0.00371 0.00314 0.02229 0.26629 0.008   0.03171 0.18514 0.00029 0.00057
1:      0.00371 0.68286 0.01286 0.07    0.02657 0.16029 0.002   0.00857 0.01598 0.03971
2:      0.00657 0.01857 0.52314 0.02543 0.00743 0.02514 0.00114 0.01029 0.19914 0.15914
3:      0.032   0.074   0.00686 0.59    0.22914 0.06514 0.03171 0.05886 0.01084 0.00771
4:      0.17086 0.01914 0.00457 0.13086 0.32371 0.02829 0.03057 0.08229 0.00114 0.00429
5:      0.01    0.14514 0.01257 0.09143 0.04486 0.62229 0.01171 0.03971 0.01854 0.02229
6:      0.02943 0.00571 0.00029 0.02543 0.04714 0.01714 0.78743 0.22029 0       0.00457
7:      0.04743 0.00229 0.002   0.02114 0.04143 0.01543 0.10029 0.378   0.00057 0.00114
8:      0.00114 0.02    0.26371 0.01086 0.004   0.03171 0.00114 0.00714 0.33837 0.35143
9:      0.004   0.02714 0.17086 0.01257 0.00943 0.02657 0.00229 0.00971 0.41512 0.40914
At 10 results
q:        0       1       2       3       4       5       6       7       8       9
r:
0:      0.69243 0.00443 0.00329 0.027   0.27129 0.00814 0.03429 0.20229 0.00043 0.00043
1:      0.00343 0.65386 0.01386 0.06971 0.02571 0.17529 0.00229 0.00971 0.02054 0.04414
2:      0.00671 0.02029 0.48714 0.02643 0.011   0.03057 0.00229 0.012   0.20029 0.16614
3:      0.03429 0.07543 0.00871 0.56971 0.22857 0.073   0.033   0.05486 0.01084 0.008
4:      0.168   0.01971 0.00457 0.13514 0.31771 0.02771 0.03757 0.09    0.001   0.00329
5:      0.00957 0.16014 0.01271 0.09114 0.04229 0.57514 0.012   0.03857 0.02382 0.02514
6:      0.02771 0.00457 0.00029 0.02957 0.04686 0.02186 0.77514 0.219   0       0.00386
7:      0.05257 0.00257 0.00243 0.02114 0.042   0.01786 0.09886 0.35514 0.00043 0.00129
8:      0.00143 0.02514 0.27843 0.01557 0.00457 0.03786 0.00129 0.00729 0.32853 0.36957
9:      0.00386 0.03243 0.18857 0.01457 0.01    0.03257 0.00329 0.01114 0.41412 0.37814
At 20 results
q:        0       1       2       3       4       5       6       7       8        9
r:
0:      0.68564 0.00557 0.0025  0.03093 0.27614 0.00807 0.0385  0.22293 0.00036 0.00057
1:      0.00357 0.62443 0.01557 0.0655  0.02393 0.18164 0.00321 0.00964 0.02575 0.05307
2:      0.00621 0.02421 0.44993 0.03043 0.01107 0.03621 0.00371 0.01343 0.20813 0.17179
3:      0.03607 0.07164 0.01064 0.53921 0.22907 0.08193 0.03407 0.05821 0.01113 0.01
4:      0.17086 0.01936 0.00507 0.14971 0.30871 0.03136 0.0375  0.09457 0.00136 0.00371
5:      0.0095  0.17007 0.01564 0.09529 0.0405  0.52879 0.01279 0.03614 0.03138 0.02743
6:      0.02521 0.00571 0.00064 0.03157 0.04879 0.02543 0.7665  0.21729 0       0.00386
7:      0.05721 0.00279 0.00264 0.02371 0.04543 0.0195  0.09821 0.32814 0.00093 0.00136
8:      0.00214 0.03257 0.29414 0.01836 0.00621 0.04907 0.00114 0.007   0.30877 0.37536
9:      0.00357 0.04221 0.20321 0.01529 0.01014 0.038   0.00436 0.01264 0.4122  0.35286
At 50 results
q:        0       1       2       3       4       5       6       7       8         9
r:
0:      0.67189 0.00734 0.00231 0.03903 0.28311 0.01006 0.04317 0.24591 0.00046 0.00083
1:      0.00431 0.57437 0.02206 0.06314 0.02377 0.18783 0.004   0.01043 0.03526 0.0726
2:      0.00586 0.02814 0.39214 0.0324  0.01249 0.04754 0.00497 0.01403 0.21347 0.17649
3:      0.03743 0.0704  0.0158  0.48871 0.22294 0.09543 0.03766 0.0632  0.01213 0.01394
```

```
4:      0.17614 0.02274 0.00609 0.16851 0.30151 0.03563 0.04263 0.10117 0.00205 0.00491
5:      0.00871 0.1768  0.01969 0.1004  0.04046 0.4598  0.01411 0.03449 0.0384  0.03337
6:      0.026   0.00551 0.00214 0.038   0.05183 0.02766 0.74557 0.21526 0.00009 0.00389
7:      0.06266 0.00363 0.00257 0.02637 0.04654 0.02063 0.10194 0.29509 0.00106 0.00211
8:      0.00297 0.04617 0.31583 0.02314 0.00651 0.06511 0.00129 0.00769 0.2877  0.38231
9:      0.00403 0.06346 0.22137 0.02029 0.01083 0.05031 0.00466 0.01274 0.40939 0.30954

---
Mean filtered genre accuracy:

At 5 results: 0.534980232321174At 10 results: 0.5132959241899322At 20 results: 0.48929874668840423At 50 results:
0.4526331852455674
---
Mean Normalised filtered genre accuracy:

At 5 results: 0.534980232321174At 10 results: 0.5132959241899322At 20 results: 0.48929874668840423At 50 results:
0.4526331852455674
---
Normalised average distance between examples of same class:
genre   0.993246508291104
artist  0.9910955823607808
trackname       1.7976931348623157E308
album   0.9979223792791547
---
Artist/Genre ratio:     0.997834449039218
---
% of files never similar at 5 results:  0.08157142857142857
% of files never similar at 10 results: 0.029857142857142856
% of files never similar at 20 results: 0.011285714285714286
% of files never similar at 50 results: 0.002428571428571428 4
---
Maximum number of times a song was similar at 5 results:       42
                Track: a003155, genre: RAPHIPHOP
Maximum number of times a song was similar at 10 results:      75
                Track: b015438, genre: METAL
Maximum number of times a song was similar at 20 results:      131
                Track: b015438, genre: METAL
Maximum number of times a song was similar at 50 results:      245
                Track: b012259, genre: METAL
---
Number of times the triangular inequality held in 200000: 200000 (100.0%)
```

Appendix 5 MIREX 2013 AMS Result (DM2)

```
Retriever: DistanceMatrixRetriever - DM2 (sparse)
Report time stamp: Fri Oct 25 12:37:16 CDT 2013

---
Neighbourhood clustering according to indexed metadatas:
            [% top 5 in same class, Random baseline, Normalised % top 5 in same class],
genre       0.7634285714285808      0.09988571428571427     0.7634285714285808
artist      0.4725999999999969      0.013046040816326553    0.4744309523809497
trackname   0.0         0.0         0.0
album       0.5494857142857114      0.05322473469387734     0.5497999999999972
            [% top 10 in same class, Random baseline, Normalised % top 10 in same class],
genre       0.6421428571428687      0.09988571428571427     0.6421428571428687
artist      0.3431142857142854      0.013046040816326553    0.35802035147392547
trackname   0.0         0.0         0.0
album       0.42220000000000274     0.05322473469387734     0.42899331065760266
            [% top 20 in same class, Random baseline, Normalised % top 20 in same class],
genre       0.5826214285714352      0.09988571428571427     0.5826214285714352
artist      0.2521571428571393      0.013046040816326553    0.30632204806225993
trackname   0.0         0.0         0.0
album       0.33867857142857205     0.05322473469387734     0.3742574257963533
            [% top 50 in same class, Random baseline, Normalised % top 50 in same class],
genre       0.5158228571428577      0.09988571428571427     0.5158228571428577
artist      0.1580742857142808      0.013046040816326553    0.2786359195350805
trackname   0.0         0.0         0.0
album       0.2503885714285635      0.05322473469387734     0.3324141555712393
---
Artist Filtered Genre Neighbourhood clustering:
                        [% top 5 in same class, Normalised % top 5 in same class]
Filtered Genre          0.5441714285714275      0.5441714285714275
                        [% top 10 in same class, Normalised % top 10 in same class]
Filtered Genre          0.5199999999999995      0.5199999999999995
                        [% top 20 in same class, Normalised % top 20 in same class]
Filtered Genre          0.4865214285714298      0.4865214285714298
                        [% top 50 in same class, Normalised % top 50 in same class]
Filtered Genre          0.42889714285713965     0.43106168344619705
---
Artist Filtered Genre Neighbourhood confusion:

Classes:
0:          METAL
1:          BLUES
2:          BAROQUE
3:          COUNTRY
4:          ROCKROLL
5:          JAZZ
6:          RAPHIPHOP
7:          EDANCE
8:          CLASSICAL
9:          ROMANTIC

At 5 results
q:      0       1       2       3       4       5       6       7       8       9
r:
0:    0.66171 0.00286 0.00314 0.024   0.218   0.00771 0.02857 0.14143 0.00057 0.00086
1:    0.004   0.65629 0.01171 0.06029 0.03371 0.16914 0.00343 0.00771 0.0097  0.01457
2:    0.002   0.02686 0.51    0.00971 0.00314 0.01371 0.00143 0.00743 0.2077  0.22029
3:    0.03914 0.08257 0.018   0.62857 0.23714 0.10771 0.03486 0.05914 0.00656 0.00629
4:    0.20771 0.02829 0.01229 0.14686 0.38914 0.03343 0.03286 0.09571 0.00143 0.00429
5:    0.01171 0.15714 0.01971 0.08457 0.04114 0.60514 0.01171 0.044   0.01455 0.00857
6:    0.02743 0.00371 0.00143 0.01971 0.04057 0.01543 0.80086 0.22743 0.00029 0.00171
7:    0.04114 0.00314 0.00429 0.01743 0.032   0.02    0.08457 0.40571 0.00057 0.00171
8:    0.002   0.014   0.22057 0.004   0.00143 0.014   0       0.00429 0.36063 0.31857
9:    0.00314 0.02371 0.19886 0.00486 0.00371 0.01371 0.00171 0.00714 0.398   0.42314
At 10 results
q:      0       1       2       3       4       5       6       7       8       9
r:
0:    0.65543 0.00371 0.00414 0.02571 0.22    0.00771 0.02643 0.15014 0.00029 0.001
1:    0.00386 0.62786 0.01557 0.06514 0.029   0.17529 0.00329 0.00829 0.0117  0.01857
2:    0.00214 0.02957 0.47457 0.01143 0.00429 0.01757 0.00357 0.00871 0.21583 0.22543
3:    0.03886 0.08514 0.02243 0.58943 0.24443 0.12229 0.03371 0.063   0.01027 0.00729
4:    0.212   0.02971 0.01214 0.16357 0.388   0.04129 0.03971 0.09814 0.002   0.00471
5:    0.011   0.16929 0.01943 0.08686 0.03771 0.55843 0.01414 0.045   0.01797 0.01157
6:    0.02814 0.00343 0.001   0.02257 0.03743 0.01929 0.78871 0.23    0.00014 0.00129
7:    0.04314 0.00386 0.00357 0.021   0.03329 0.02071 0.08886 0.38314 0.00128 0.00143
8:    0.002   0.01614 0.23657 0.00714 0.00171 0.02071 0       0.00571 0.34194 0.33671
9:    0.00343 0.02986 0.21057 0.00714 0.00414 0.01671 0.00157 0.00786 0.39857 0.392
At 20 results
q:      0       1       2       3       4       5       6       7       8       9
r:
0:    0.63557 0.00407 0.00336 0.02693 0.2295  0.0085  0.02786 0.15764 0.00021 0.001
1:    0.00436 0.59043 0.0205  0.065   0.0275  0.1875  0.00321 0.00793 0.01633 0.02621
2:    0.00329 0.03479 0.42936 0.01729 0.00621 0.02629 0.0055  0.00993 0.22611 0.22957
3:    0.04393 0.08193 0.03257 0.54679 0.24564 0.14171 0.03607 0.06943 0.01434 0.01043
4:    0.2205  0.03007 0.01236 0.18214 0.37364 0.04614 0.04429 0.10736 0.00243 0.00579
5:    0.01136 0.18857 0.02307 0.09271 0.038   0.49821 0.01357 0.04229 0.02361 0.01643
6:    0.02786 0.00321 0.00129 0.02407 0.03714 0.02114 0.76764 0.22907 0.00043 0.00214
7:    0.04764 0.005   0.00371 0.02571 0.0355  0.02186 0.09929 0.36157 0.00164 0.002
8:    0.00229 0.02329 0.24814 0.01    0.00207 0.02893 0.00021 0.00607 0.3169  0.36179
9:    0.00321 0.03721 0.22564 0.00936 0.00479 0.01971 0.00236 0.00871 0.398   0.34464
At 50 results
q:      0       1       2       3       4       5       6       7       8       9
r:
0:    0.58957 0.00514 0.00343 0.03863 0.24031 0.01003 0.03474 0.16923 0.00086 0.00177
1:    0.00449 0.5244  0.03331 0.06266 0.02503 0.19566 0.00277 0.00889 0.03039 0.05211
2:    0.00483 0.04234 0.36251 0.02746 0.01011 0.04209 0.00663 0.01109 0.21244 0.23291
3:    0.053   0.078   0.04931 0.46354 0.2402  0.1524  0.04414 0.0762  0.01946 0.02009
```

```
4:    0.23957 0.02929 0.01694 0.20803 0.35137 0.0512  0.04766 0.11409 0.00417 0.00646
5:    0.01063 0.21717 0.03066 0.10469 0.03837 0.41614 0.01529 0.03797 0.03692 0.02877
6:    0.03166 0.00369 0.00489 0.0324  0.04183 0.02631 0.7302  0.22831 0.00071 0.00426
7:    0.06083 0.00597 0.00594 0.0306  0.0438  0.02557 0.115   0.3386  0.00225 0.00634
8:    0.00249 0.0374  0.26023 0.01709 0.00349 0.0456  0.00109 0.00626 0.24374 0.37851
9:    0.00294 0.05517 0.23223 0.01491 0.00549 0.035   0.00249 0.00937 0.3598  0.26854

---
Mean filtered genre accuracy:

At 5 results: 0.5441199103321785At 10 results: 0.5199511514163435At 20 results: 0.48647615651110643At 50 results:
0.42886232321173823
---
Normalised average distance between examples of same class:
genre        0.9932168836613334
artist       0.984138566332805
trackname    1.7976931348623157E308
album        0.9928682568161565
---
Artist/Genre ratio:      0.9908596828367812
---
% of files never similar at 5 results:0.06942857142857142
% of files never similar at 10 results:            0.029
% of files never similar at 20 results:            0.007714285714285714
% of files never similar at 50 results:            0.001
---
Maximum number of times a song was similar at 5 results:        24
                          Track: a001629, genre: RAPHIPHOP
Maximum number of times a song was similar at 10 results:       49
                          Track: a001629, genre: RAPHIPHOP
Maximum number of times a song was similar at 20 results:       74
                          Track: a004535, genre: RAPHIPHOP
Maximum number of times a song was similar at 50 results:       153
                          Track: a005185, genre: RAPHIPHOP
---
```

Number of times the triangular inequality held in 200000: 200000 (100.0%)

Bibliography

Aliyev, Roman. 2013. “Using Continuous Wavelet Transform and Hidden Markov Models For Audio Music Similarity Estimation.” In *Submission to Audio Music Similarity and Retrieval Task of MIREX 2013*, [Online]. Available: www.music-ir.org/mirex/abstracts/2013/RA1.pdf

Alluri, Vinoo, and Petri Toiviainen. 2009. “Exploring Perceptual and Acoustical Correlates of Polyphonic Timbre.” *Music Perception* 27 (3): 223–242.

Alonso, Miguel, and Bertrand David. 2004. “Tempo and Beat Estimation of Musical Signals.” In *Proceedings of 5th International Society for Music Information Retrieval Conference*, 158–163. Barcelona, Spain.

Anglade, Amelie, Rafael Ramirez, and Simon Dixon. 2009. “Genre Classification Using Harmony Rules Induced from Automatic Chord Transcriptions.” In *Proceedings of 10th International Society for Music Information Retrieval Conference*, 669–674.

Arthur, David, and Sergei Vassilvitskii. 2007. “K-Means ++ : The Advantages of Careful Seeding.” In *SODA ’07 Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, 1027–1035. New Orleans, Louisiana.

Aucouturier, Jean-Julien, and Emmanuel Bigand. 2013. “Seven Problems That Keep MIR from Attracting the Interest of Cognition and Neuroscience.” *Journal of Intelligent Information Systems* 41 (3) (July 5): 483–497.

Aucouturier, Jean-Julien, and François Pachet. 2002. “Music Similarity Measures : What’s the Use?” In *3rd International Conference on Music Information Retrieval*, 157–163. Paris, France.

Aucouturier, Jean-Julien, and François Pachet. 2004. “Improving Timbre Similarity : How High’s the Sky?” *J. Negative Results Speech Audio Sci.* 1 (1).

Aucouturier, Jean-Julien, François Pachet, and Mark Sandler. 2005. “‘The Way It Sounds’: Timbre Models for Analysis and Retrieval of Music Signals.” *IEEE Transactions on Multimedia* 7 (6) (December): 1028–1035.

Barthet, Mathieu, György Fazekas, and Mark Sandler. 2013. “Music Emotion Recognition : From Content- to Context-Based Models.” In *From Sounds to Music and Emotions*, 228–252. Springer Berlin Heidelberg.

Barutcuoglu, Zafer, Robert Schapire, Olga G Troyanskaya, and Christopher R Decoro. 2007. “Bayesian Aggregation for Hierarchical Classification.” In *Proceedings of the International Conference on Music Information Retrieval*, 77–80.

Beckmann, Norbert, Hans-Peter Kriegel, Ralf Schneider, and Bernhard Seeger. 1990. "The R*-Tree: An Efficient and Robust Access Method for Points and Rectangles." In *Proceedings of the 1990 ACM SIGMOD International Conference on Management of Data - SIGMOD '90*, 322–331. New York, New York, USA: ACM Press.

Bellman, Richard. 1961. *Adaptive Control Processes: A Guided Tour*. Princeton, N.J.: Princeton University Press.

Benetos, Emmanouil, Simon Dixon, Dimitrios Giannoulis, Holger Kirchhoff, and Anssi Klapuri. 2013. "Automatic Music Transcription: Challenges and Future Directions." *Journal of Intelligent Information Systems* 41 (3): 407–434.

Bengio, Yoshua, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. 2007. "Greedy Layer-Wise Training of Deep Networks." In *Advances in Neural Information Processing Systems 19*, 153–160. MIT Press.

Berenzweig, Adam, Daniel Ellis, and Steve Lawrence. 2003. "Anchor Space for Classification and Similarity Measurement of Music." In *2003 International Conference on Multimedia and Expo. ICME '03. Proceedings (Cat. No.03TH8698)*, I–29. Baltimore, Maryland.

Berger, Kenneth W. 1964. "Some Factors in the Recognition of Timbre." *The Journal of the Acoustical Society of America* 36 (10): 1888.

Bishop, Christopher. 2007. *Pattern Recognition and Machine Learning*. Springer.

Blume, Friedrich. 1972. *Classic and Romantic Music : A Comprehensive Survey*. London: Faber.

Boser, Bernhard, Isabelle Guyon, and Vladimir Vapnik. 1992. "A Training Algorithm for Optimal Margin Classifiers." In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory - COLT '92*, 144–152. New York, New York, USA: ACM Press.

Brown, Judith, and Miller Puckette. 1992. "An Efficient Algorithm for the Calculation of a Constant Q Transform." *J. Acoust. Soc. Amer.* 92. (5): 2698–2701.

Cabero, Jose Maria, Fernando De la Torre, Galder Unibaso, and Aritz Sanchez. 2008. "Tracking Algorithms Based on Dynamics of Individuals and Multidimensional Scaling." In *2008 3rd International Symposium on Wireless Pervasive Computing*, 388–395.

Cano, Pedro, Eloi Batlle, Ton Kalker, and Jaap Haitsma. 2005. "A Review of Audio Fingerprinting." *The Journal of VLSI Signal Processing-Systems for Signal, Image, and Video Technology* 41 (3): 271–284.

Cano, Pedro, Martin Kaltenbrunner, Fabien Gouyon, and Eloi Batlle. 2002. "On the Use of FastMap for Audio Retrieval and Browsing." In *ISMIR 2002*, Paris, France.

Casey, Michael, and Malcolm Slaney. 2006. "The Importance of Sequences in Musical Similarity." In *2006 IEEE International Conference on Acoustics Speed and Signal Processing Proceedings*, 5:V–5–V–8.

Casey, Michael, Remco Veltkamp, Masataka Goto, Marc Leman, Christophe Rhodes, and Malcolm Slaney. 2008. "Content-Based Music Information Retrieval: Current Directions and Future Challenges." *Proceedings of the IEEE* 96 (4): 668–696.

Chang, Chih-Chung, and Chih-Jen Lin. 2011. "LIBSVM." *ACM Transactions on Intelligent Systems and Technology* 2 (3): 1–27.

Charbuillet, Christophe, Geoffroy Peeters, Stanislav Barton, and Valerie Gouet-Brunet. 2010. "A Fast Algorithm for Music Search by Similarity in Large Databases Based on Modified Symetrized Kullback Leibler Divergence." In *2010 International Workshop on Content Based Multimedia Indexing (CBMI)*, 1–6.

Ciaccia, Paolo, Marco Patella, and Pavel Zezula. 1997. "M-Tree : An Efficient Access Method for Similarity Search in Metric Spaces." In *Proceedings of the 23rd International Conference on Very Large Data Bases*, 426–435. San Francisco, Ca, USA.

Cleverdon, Cyril, and Michael Keen. 1966. "Factors Determining the Performance of Indexing Systems." *ASLIB Cranfield Research Project*.

Cohen, William. 2000. "Web-Collaborative Filtering: Recommending Music by Crawling the Web." *Computer Networks* 33 (1-6): 685–698.

Cooley, James, and John Tukey. 1965. "An Algorithm for the Machine Calculation of Complex Fourier Series." *Mathematics of Computation* 19 (90): 297–301.

Crawford, Richard. 2001. *An Introduction to America's Music*. New York: W. W. Norton.

Cunningham, Sally Jo, and David Bainbridge. 2010. "A Search Engine Log Analysis of Music-Related Web Searching." In *Advances in Intelligent Information and Database Systems*, 79–88. Springer Berlin Heidelberg.

Datar, Mayur, Nicole Immorlica, Piotr Indyk, and Vahab S. Mirrokni. 2004. "Locality-Sensitive Hashing Scheme Based on P-Stable Distributions." In *Proceedings of the Twentieth Annual Symposium on Computational Geometry - SCG '04*, 253. New York, New York, USA: ACM Press.

Davis, Steven, and Paul Mermelstein. 1980. "Comparison of Parametric Representations for Monosyllabic Word Recognition in Continuously Spoken Sentences." *IEEE Transactions on Acoustics, Speech, and Signal Processing* 28 (4): 357–366.

De, Debashis, and Samarjit Roy. 2012. "Inheritance in Indian Classical Music: An Object-Oriented Analysis and Pattern Recognition Approach." In *2012 International Conference on Radar, Communication and Computing (ICRCC)*, 193–198.

De Leon, Franz, and Kirk Martinez. 2012a. "Enhancing Timbre Model Using MFCC and Its Time Derivatives for Music Similarity Estimation." In *Proc. of the 20th European Signal Processing Conference*, 2005–2009. Bucharest, Romania.

De Leon, Franz, and Kirk Martinez. 2012b. "Towards Efficient Music Genre Classification Using FastMap." In *Proc. of the 15th International Conference on Digital Audio Effects*, 385–388. York, United Kingdom.

De Leon, Franz, and Kirk Martinez. 2012c. "A Music Genre Classifier Combining Timbre, Rhythm and Tempo Models." In *TENCON 2012 IEEE Region 10 Conference*, 1–5, Cebu, Philippines.

De Poli, Giovanni, and Paolo Prandoni. 1997. "Sonological Models for Timbre Characterization." *Journal of New Music Research* 26 (2): 170–197.

Dixon, Simon. 2001. "Automatic Extraction of Tempo and Beat From Expressive Performances." *Journal of New Music Research* 30 (1): 39–58.

Downie, J Stephen. 2005. "Music Information Retrieval." *Annual Review of Information Science and Technology* 35 (10): 83–340.

Downie, J Stephen. 2008. "The Music Information Retrieval Evaluation Exchange (2005–2007): A Window into Music Information Retrieval Research." *Acoustical Science and Technology* 29 (4): 247–255.

Downie, J Stephen, Tim Crawford, and Donald Byrd. 2009. "Ten Years of ISMIR: Reflections on Challenges and Opportunities." In *Proceedings of 10th International Society for Music Information Retrieval Conference*, 13–18.

Edwards, Michael. 2011. "Algorithmic Composition: Computational Thinking in Music." *Communications of the ACM*, 54 (7): 56-67.

Ellis, Daniel, and Graham Poliner. 2007. "Identifying `Cover Songs' with Chroma Features and Dynamic Programming Beat Tracking." In *2007 IEEE International Conference on Acoustics, Speech and Signal Processing - ICASSP '07*, IV–1429–IV–1432.

Faloutsos, Christos, and King-Ip Lin. 1995. "FastMap." *ACM SIGMOD Record* 24 (2): 163–174.

Fastl, Hugo. 2005. "Psycho-Acoustics and Sound Quality." In *Communication Acoustics*, 139–162. Springer Berlin Heidelberg.

Fastl, Hugo, and Eberhard Zwicker. 2006. *Psychoacoustics Facts and Models (Springer Series in Information Sciences). Springer Series in Information Sciences*. 3rd ed. Springer.

Fingerhut, Michael. 2004. "Music Information Retrieval , or How to Search for (and Maybe Find) Music and Do Away with Incipits." Slides for *IAML-IASA Congress*. Olso. [Online]. Available: http://articles.ircam.fr/textes/Fingerhut04b/note.pdf

Flexer, Arthur, Elias Pampalk, and Gerhard Widmer. 2005. "Hidden Markov Models for Spectral Similarity of Songs." In *Proc. of the 8th Int. Conference on Digital Audio Effects (DAFx'05)*. Madrid, Spain.

Foote, Jonathan T. 1997. "Content-Based Retrieval of Music and Audio." In *Proceedings of SPIE*, 3229: 138–147.

Fujishima, Takuya. 1999. "Realtime Chord Recognition of Musical Sound: A System Using Common Lisp Music." In *Proc. of International Computer Music Conference*, 464–467. Beijing, China.

Furui, Sadaoki. 1986. "Speaker-Independent Isolated Word Recognition Using Dynamic Features of Speech Spectrum." *IEEE Transactions on Acoustics, Speech, and Signal Processing* 34 (1): 52–59.

Gargantini, Irene. 1982. "An Effective Way to Represent Quadtrees." *Communications of the ACM* 25 (12): 905–910.

Gjerdingen, Robert, and David Perrott. 2008. "Scanning the Dial: The Rapid Recognition of Music Genres." *Journal of New Music Research* 37 (2): 93–100.

Grey, John M. 1975. "An Exploration of Musical Timbre". *Doctoral Dissertation*. Stanford University.

Grey, John M. 1977. "Multidimensional Perceptual Scaling of Musical Timbres." *The Journal of the Acoustical Society of America* 61 (5): 1270.

Grey, John M. 1978. "Perceptual Effects of Spectral Modifications on Musical Timbres." *The Journal of the Acoustical Society of America* 63 (5): 1493.

Grimaldi, Marco, Pádraig Cunningham, and Anil Kokaram. 2003. "A Wavelet Packet Representation of Audio Signals for Music Genre Classification Using Different Ensemble and Feature Selection Techniques." In *Proceedings of the 5th ACM SIGMM International Workshop on Multimedia Information Retrieval - MIR '03*, 102. New York, New York, USA: ACM Press.

Gruzd, Anatoliy A, J Stephen Downie, M Cameron Jones, and Jin Ha Lee. 2007. "Evalutron 6000." In *Proceedings of the 2007 Conference on Digital Libraries - JCDL '07*, 1:507. New York, New York, USA: ACM Press.

Guttman, Antonin. 1984. "R-Trees: A Dynamic Index Structure for Spatial Searching." *ACM SIGMOD Record* 14 (2): 47.

Hall, Joseph. 2013. "Application of Multidimensional Scaling to Subjective Evaluation of Coded Speech." In *2000 IEEE Workshop on Speech Coding. Proceedings. Meeting the Challenges of the New Millennium (Cat. No.00EX421)*, 20–22.

Hamel, Philippe, and Douglas Eck. 2010. "Learning Features from Music Audio with Deep Belief Networks." In *Proc. of the 11th International Society for Music Information Retrieval Conference (ISMIR 2010)*, 339–344.

Hamel, Philippe, Sean Wood, and Douglas Eck. 2009. "Automatic Identification of Instrument Classes in Polyphonic and Poly-Instrument Audio." In *Proc. of 10th International Society for Music Information Retrieval Conference*. Kobe, Japan.

Hennessy, John, and David Patterson. 2006. *Computer Architecture : A Quantitative Approach*. 4th Editio. Burlington, MA, USA: Morgan Kaufmann.

Herlocker, Jonathan, Joseph Konstan, Al Borchers, and John Riedl. 1999. "An Algorithmic Framework for Performing Collaborative Filtering. " In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval - SIGIR '99*. New York, New York, USA: ACM Press.

Hesmondhalgh, David, and Keith Negus. 2002. *Popular Music Studies*. London: Bloomsbury Academic.

Hill, John Walter. 2005. *Baroque Music : Music in Western Europe, 1580-1750*. New York: W.W. Norton & Company.

Hinrichs, Klaus. 1985. "Implementation of the Grid File: Design Concepts and Experience." *BIT Numerical Mathematics* 25 (4): 569–592.

Hjaltason, Gisli, and Hanan Samet. 2003. "Index-Driven Similarity Search in Metric Spaces." *ACM Transactions on Database Systems* 28 (4): 517–580.

Hjaltason, Gisli, and Hanan Samet. 2003. "Properties of Embedding Methods for Similarity Searching in Metric Spaces." *IEEE Transactions on Pattern Analysis and Machine Intelligence* 25 (5): 530–549.

Houtsma, Adrianus. 1997. "Pitch and Timbre: Definition, Meaning and Use." *Journal of New Music Research* 26 (2): 104–115.

Hsu, Chih-Wei, and Chin-Jen Lin. 2002. "A Comparison of Methods for Multiclass Support Vector Machines." *IEEE Transactions on Neural Networks* 13 (4): 1026–7.

Humphrey, Eric, Juan Bello, and Yann LeCun. 2013. "Feature Learning and Deep Architectures: New Directions for Music Informatics." *Journal of Intelligent Information Systems* 41 (3): 461–481.

Humphrey, Eric, Juan Pablo Bello, and Yann LeCun. 2012. "Moving Beyond Feature Design: Deep Architectures and Automatic Feature Learning in Music Informatics." In *Proc. of 13th International Society for Music Information Retrieval Conference*, 403–408. Porto, Portugal.

IFPI. 2013. "IFPI Digital Music Report 2013 Engine of a Digital World." [Online]. Available: http://www.ifpi.org/content/section_resources/dmr2013.html.

"ISMIR 2004 Audio Description Contest-Genre/Artist ID Classification and Artist Similarity." [Online] Available: http://ismir2004.ismir.net/genre_contest/index.htm.

Iverson, Paul. 1993. "Isolating the Dynamic Attributes of Musical Timbre." *The Journal of the Acoustical Society of America* 94 (5): 2595.

Jang, Jyh-Shing Roger. 2011. "Combining Visual and Acoustic Features for Music Genre Classification." In *2011 10th International Conference on Machine Learning and Applications and Workshops*, 124–129.

Jegou, Hervé, Cordelia Schmid, Hedi Harzallah, and Jakob Verbeek. 2010. "Accurate Image Search Using the Contextual Dissimilarity Measure." *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32 (1): 2–11.

Jensen, Jesper Højvang, Mads Græsbøll Christensen, Manohar Murthi, and Søren Holdt Jensen. 2006. "Evaluation of MFCC Estimation Techniques for Music Similarity." In *Proceedings of the 14th European Signal Processing Conference*. Florence, Italy.

Jiang, Dan-Ning, Lie Lu, Hong-jiang Zhang, Jian-hua Tao, and Lian-hong Cui. 2002. "Music Type Classification by Spectral Contrast Feature." In *Proceedings. IEEE International Conference on Multimedia and Expo*, 113–116.

Jones, Cameron, J Stephen Downie, and Andreas Ehmann. 2007. "Human Similarity Judgments: Implications for the Design of Formal Evaluations." In *Proceedings of International Society for Music Information Retrieval Conference*, 1–4.

Juslin, Patrik, and Petri Laukka. 2004. "Expression, Perception, and Induction of Musical Emotions: A Review and a Questionnaire Study of Everyday Listening." *Journal of New Music Research* 33 (3): 217–238.

Kamel, Ibrahim, and Christos Faloutsos. 1993. "Hilbert R-Tree: An Improved R-Tree Using Fractals". Department of Computer Science, University of Maryland.

Karydis, Ioannis, Milos Radovanovic, Alexandros Nanopoulos, and Mirjana Ivanovic. 2010. "Looking Through the 'Glass Ceiling': A Conceptual Framework for the Problems of Spectral Similarity." In *Proceedings of 11th International Society for Music Information Retrieval ConferenceInformation Retrieval*, 267–272. Utrecht.

Kassler, Michael. 1966. "Computer Forum : Toward Musical Information." *Perspectives of New Music* 4 (2): 59–67.

Kautz, Henry, Bart Selman, and Mehul Shah. 1997. "Referral Web: Combining Social Networks and Collaborative Filtering." *Communications of the ACM* 40 (3): 63–65.

Kingman, Daniel. 1990. *American Music: A Panorama*. 2nd ed. New York: Schirmer Books.

Klapuri, Anssi. 1999. "Sound Onset Detection by Applying Psychoacoustic Knowledge." In *1999 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings. ICASSP99 (Cat. No.99CH36258)*, 6: 3089–3092.

Korn, Flip, Nikolaos Sidiropoulos, Christos Faloutsos, Eliot Siegel, and Zenon Protopapas. 1996. "Fast Nearest Neighbor Search Databases in Medical Image Databases." In *Proceedings of the Int. Conf. on Very Large Data Bases*, 215–226.

Krimphoff, Jochen, Stephen McAdams, and Suzanne Winsberg. 1994. "Caractérisation Du Timbre Des Sons complexes.II. Analyses Acoustiques et Quantification Psychophysique." *Le Journal de Physique IV* 04 (C5): C5–625–C5–628.

Krumhansl, Carol. 2002. "Music : A Link Between Cognition and Emotion." *Current Directions in PSychological Science* 11 (2): 45–50.

Kruskal, Joseph. 1964. "Multidimensional Scaling by Optimizing Goodness of Fit to a Nonmetric Hypothesis." *Psychometrika* 29 (1): 1–27.

Kullback, Solomon, and Richard Leibler. 1951. "On Information and Sufficiency." *The Annals of Mathematical Statistics* 22 (1): 79–86.

Lartillot, Olivier, and Petri Toiviainen. 2007. "A Matlab Toolbox for Musical Feature Extraction From Audio." In *Proceedings of the 10th International Conference on Digital Audio Effects (DAFx-07)*, 1–8. Bordeaux, France.

Li, Ming, and Ronan Sleep. 2005. "Genre Classification via an LZ78-Based String Kernel." In *Proceedings of 6th International Society for Music Information Retrieval Conference*, London, United Kingdom.

Li, Tao, and Mitsunori Ogihara. 2005. "Music Genre Classification with Taxonomy." In *Proceedings. (ICASSP '05). IEEE International Conference on Acoustics, Speech, and Signal Processing, 2005.*, 5: 197–200.

Li, Tao, Mitsunori Ogihara, and Qi Li. 2003. "A Comparative Study on Content-Based Music Genre Classification." In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval - SIGIR '03*, 282. New York, New York, USA: ACM Press.

Liu, Dan, Lie Lu, and Hong-Jiang Zhang. 2003. "Automatic Mood Detection from Acoustic Music Data." In *Proceedings of International Symposium on Music Information Retrieval*, 81–87.

Liu, Xiaobing, Deshun Yang, and Xiaoou Chen. 2008. "New Approach to Classification of Chinese Folk Music Based on Extension of HMM." In *2008 International Conference on Audio, Language and Image Processing*, 1172–1179.

Logan, Beth. 2000. "Mel Frequency Cepstral Coefficients for Music Modeling." In *Proc. Int. Sym. Music Information Retrieval (ISMIR)*, Massachusetts, USA.

Logan, Beth, and Ariel Salomon. 2001. "A Music Similarity Function Based on Signal Analysis." In *IEEE International Conference on Multimedia and Expo, 2001. ICME 2001.*, 745–748.

Longley, Paul, Michael Goodchile, David Maguire, and David Rhind, ed. 1999. "Spatial Access Methods." In *Geographical Information Systems: Principles, Techniques, Management and Applications, 2nd Edition, Abridged*, 1:385–400. Wiley.

Lukashevich, Hanna, Jakob Abeßer, Christian Dittmar, and Holger Grossmann. 2009. "From Multi-Labeling to Multi-Domain-Labeling : A Novel Two-Dimensional Approach to Music Genre Classification." In *Proceedings of 10th International Society for Music Information Retrieval Conference*, 459–464.

Macqueen, James. 1967. "Some Methods for Classification and Analysis of Multivariate Observations." In *Proc. Fifth Berkeley Symposium on Mathematical Statististics and Probability*, 233: 281–297.

Mandel, Michael, and Daniel Ellis. 2005. "Song-Level Features and Support Vector Machines for Music Classification." In *Submission to MIREX 2005*, 594–599.

McAdams, Stephen, Suzanne Winsberg, Sophie Donnadieu, Geert Soete, and Jochen Krimphoff. 1995. "Perceptual Scaling of Synthesized Musical Timbres: Common Dimensions, Specificities, and Latent Subject Classes." *Psychological Research* 58 (3): 177–192.

McFee, Brian, Thierry Bertin-Mahieux, Daniel Ellis, and Gert Lanckriet. 2012. "The Million Song Dataset Challenge." In *Proceedings of the 21st International Conference Companion on World Wide Web - WWW '12 Companion*, 909. New York, New York, USA: ACM Press.

Mckay, Cory, and Ichiro Fujinaga. 2004. "Automatic Genre Classification Using High-Level Musical Feature Sets." In *Proc. Int. Sym. Music Information Retrieval (ISMIR),* Barcelona, Spain.

Meng, Anders., Peter Ahrendt, and Jan Larsen. 2005. "Improving Music Genre Classification By Short-Time Feature Integration." In *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing 2005*, 5: 497–500.

Meng, Anders, and John Shawe-Taylor. 2005. "An Investigation of Feature Models for Music Genre Classification Using the Support Vector Classifier." In *Proceedings of 6th International Society for Music Information Retrieval Conference*, 604–609.

Meyer, Scott, and Patrick Wolf. 1997. "Application of Sonomicrometry and Multidimensional Scaling to Cardiac Catheter Tracking." *IEEE Transactions on Bio-Medical Engineering* 44 (11): 1061–7.

Miller, Robert. 1968. "Response Time in Man-Computer Conversational Transactions." In *Proceedings of the December 9-11, 1968, Fall Joint Computer Conference, Part I on - AFIPS '68 (Fall, Part I)*, 267. New York, New York, USA: ACM Press.

Ono, Nobutaka, Kenichi Miyamoto, Hirokazu Kameoka, and Shigeki Sagayama. 2008. "A Real-Time Equalizer of Harmonic and Percussive Components in Music Signals." In *Proc. of International Society for Music Information Retrieval Conference 2008*, 139–144.

Oppenheim, Alan, and Ronald Schafer. 1989. *Discrete-Time Signal Processing*. Upper Saddle River, New Jersey, USA: Prentice-Hall, Inc.

Orenstein, Jack, and Frank Manola. 1988. "PROBE Spatial Data Modeling and Query Processing in an Image Database Application." *IEEE Transactions on Software Engineering* 14 (5): 611–629.

Orenstein, Jack. 1989. "Redundancy in Spatial Databases." *ACM SIGMOD Record* 18 (2): 295–305.

Orio, Nicola. 2006. "Music Retrieval: A Tutorial and Review." *Foundations and Trends® in Information Retrieval* 1 (1): 1–96.

Orpen, Keith, and David Huron. 1992. "Measurements of Similarity in Music: A Quantitative Approach for Non-Parametric Representations." *Computers in Music Research* 4 (Fall): 1–44.

Pampalk, Elias. 2006a. "Audio-Based Music Similarity and Retrieval : Combining a Spectral Similarity Model with Information Extracted from Fluctuation Patterns." In *Submission to MIREX 2006*.

Pampalk, Elias. 2006b. "Computational Models of Music Similarity and Their Application in Music Information Retrieval." *Doctoral Dissertation*. Vienna University of Technology.

Pampalk, Elias, Simon Dixon, and Gerhard Widmer. 2004. "Exploring Music Collections by Browsing Different Views." *Computer Music Journal* 28 (2): 49–62.

Pampalk, Elias, Arthur Flexer, and Gerhard Widmer. 2005. "Improvements of Audio-Based Music Similarity and Genre Classification." In *Proceedings of ISMIR 2005 Sixth International Conference on Music Information Retrieval*, Barcelona, Spain.

Pampalk, Elias, Andreas Rauber, and Dieter Merkl. 2002. "Content-Based Organization and Visualization of Music Archives." In *Proceedings of the Tenth ACM International Conference on Multimedia - MULTIMEDIA '02*, 570. New York, New York, USA: ACM Press.

Penny, William. 2001. "Kullback-Leibler Divergences of Normal, Gamma, Dirichlet and Wishart Densities." *Wellcome Department of Cognitive Neurology*. [Online] Available: www.fil.ion.ucl.ac.uk/~wpenny/publications/densities.ps

Peretz, Isabelle. 1998. "Music and Emotion: Perceptual Determinants, Immediacy, and Isolation after Brain Damage." *Cognition* 68 (2): 111–141.

Platt, John, Nello Cristianini, and John Shawe-taylor. 2000. "Large Margin DAGs for Multiclass Classification." *Advances in Neural Information Processing Systems* 12: 547–553.

Plomp, Reinier. 1970. "Timbre as a Multidimensional Attribute of Complex Tones." In *Frequency Analysis and Periodicity Detection in Hearing*, 397–414.

Pohle, Tim, Dominik Schnitzer, Markus Schedl, Peter Knees, and Gerhard Widmer. 2009. "On Rhythm and General Music Similarity." In *Submission to Audio Music Similarity and Retrieval Task of MIREX 2009*, 525–530.

Polotti, Pietro, and Davide Rocchesso. 2008. *Sound to Sense , Sense to Sound A State of the Art in Sound and Music Computing*. Edited by Pietro Polotti and Davide Rocchesso. *Framework*. Firenze.

Proakis, John, and Dimitris Manolakis. 2006. *Digital Signal Processing*. Prentice Hall.

Pye, David. 2000. "Content-Based Methods for the Management of Digital Music." In *2000 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No.00CH37100)*, 4: 2437–2440. Istanbul, Turkey.

Rabiner, Lawrence, and Biing-Hwang Juang. 1986. "An Introduction to Hidden Markov Models." *IEEE ASSP Magazine*, 3 (1): 4-16.

Rabiner, Lawrence, and Biing-Hwang Juang. 1993. *Fundamentals of Speech Recognition*. 1st ed. Upper Saddle River, New Jersey, USA: Prentice-Hall, Inc.

Radovanovic, Milos, Alexandros Nanopoulos, and Mirjana Ivanovic. 2010. "On the Existence of Obstinate Results in Vector Space Models." In *Proceeding of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval - SIGIR '10*, 186. New York, New York, USA: ACM Press.

Reed, Jeremy, and Chin-hui Lee. 2006. "A Study on Music Genre Classification Based on Universal Acoustic Models." *Proceedings of 7th International Society for Music Information Retrieval Conference*: 89–94.

Rioul, Olivier, and Martin Vetterli. 1991. "Wavelets and Signal Processing." *IEEE Signal Processing Magazine* 8 (4): 14–38.

Rosoff, Matt. 2009. "Mufin Player Organizes Songs by Sound _ Digital Noise Music and Tech - CNET News." [Online] Available: http://news.cnet.com/8301-13526_3-10184456-27.html.

Rovi. 2013. "Music Database Metadata - All Music Guide Data - Rovi Music Data." [Online] Available: http://www.rovicorp.com/products/discovery/metadata/music-data.htm?tabContent=/products/discovery/metadata/rovi-music/edition-13178.htm.

Rubner, Yossi, Carlo Tomasi, and Leonidas Guibas. 1998. "A Metric for Distributions with Applications to Image Databases." In *Sixth International Conference on Computer Vision (IEEE Cat. No.98CH36271)*, 59–66. Bombay, India: Narosa Publishing House.

Russell, James. 1980. "A Circumplex Model of Affect." *Journal of Personality and Social Psychology* 39 (6): 1161–1178.

Salton, Gerard. 1992. "The State of Retrieval System Evaluation." *Information Processing & Management* 28 (4): 441–449.

Schmidt, Erik, Jeffrey Scott, and Youngmoo Kim. 2012. "Feature Learning in Dynamic Environments: Modeling the Coustic Structure of Musical Emotion." In *Proc. of 13th International Society for Music Information Retrieval Conference*, 325–330. Porto, Portugal.

Schnitzer, Dominik, Arthur Flexer, Markus Schedl, and Gerhard Widmer. 2012. "Local and Global Scaling Reduce Hubs in Space." *Journal of Machine Learning* 13: 2871–2902.

Schnitzer, Dominik, Arthur Flexer, and Gerhard Widmer. 2009. "A Filter-and-Refine Method for Fast Similarity Search in Millions of Tracks." In *ISMIR 2009*, 537–542.

Schouten, Jan. 1968. "The Perception of Timbre." In *Reports of the 6th International Congress on Acoustics*, 35–44. Tokyo.

Schwarz, Diemo. 1998. "Spectral Envelopes in Sound Analysis and Synthesis". *Doctotal Dissertation.* Universitat Stuttgart.

Seidl, Thomas, and Hans-Peter Kriegel. 1998. "Optimal Multi-Step K-Nearest Neighbor Search." In *Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data - SIGMOD '98*, 27:154–165. New York, New York, USA: ACM Press.

Sellis, Timos, Nick Roussopoulos, and Christos Faloutsos. 1987. "The R+-Tree: A Dynamic Index for Multi-Dimensional Objects." In *Proc. of 13th Intenational Conference on VLDB*, 517–518. England.

Serra, Joan, and Emilia Gomez. 2008. "Audio Cover Song Identification Based on Tonal Sequence Alignment." In *2008 IEEE International Conference on Acoustics, Speech and Signal Processing*, 61–64.

Seyerlehner, Klaus, Markus Schedl, Tim Pohle, and Peter Knees. 2010. "Using Block-Level Features for Genre Classification , Tag Classification and Music Similarity Estimation." In *Submission to Audio Music Similarity and Retrieval Task of MIREX 2010*. Linz, Austria.

Seyerlehner, Klaus, Gerhard Widmer, and Peter Knees. 2008. "Frame Level Audio Similarity - A Codebook Approach." In *Proc. of the 13th Conference on DIgital Audio Effects*, 1–8. Espoo, Finland.

Seyerlehner, Klaus, Gerhard Widmer, and Tim Pohle. 2010. "Fusing Block-Level Features for Music Similarity Estimation." In *Proc. of the 11th Conference on DIgital Audio Effects*, 1–8. Graz, Austria.

Seyerlehner, Klaus, Gerhard Widmer, Markus Schedl, and Peter Knees. 2010. "Automatic Music Tag Classification Based on Block-Level." In *Proceedings of Sound and Music Computing 2010*, Barcelona, Spain.

Shannon, Claude. 1948. "A Mathematical Theory of Communication." *The Bell System Technical Journal* 27 (3): 379–423.

Shepard, Roger. 1962a. "The Analysis of Proximities: Multidimensional Scaling with an Unknown Distance Function. I." *Psychometrika* 27 (2): 125–140.

Shepard, Roger. 1962b. "The Analysis of Proximities: Multidimensional Scaling with an Unknown Distance Function. II." *Psychometrika* 27 (3): 219–246.

Shlens, Jonathon. 2009. "A Tutorial on Principal Component Analysis." [Online]. Available: http://www.cs.cmu.edu/~elaw/papers/pca.pdf

Skopal, Tomas, Jaroslav Pokorn, and Vaclav Snasel. 2004. "PM-Tree : Pivoting Metric Tree for Similarity Search in Multimedia Databases." In *Proc. of Conference on Advances in Databases and Information Systems*. Budapest, Hungary.

Smalley, Denis. 1994. "Defining Timbre — Refining Timbre." *Contemporary Music Review* 10 (2): 35–48.

Smith, Jordan, Ashley Burgoyne, Ichiro Fujinaga, David De Roure, and J Stephen Downie. 2011. "Design and Creation of a Large-Scale Database of Structural Annotations." In *Proc. of International Society for Music Information Retrieval Conference*, 555–560. Miami, USA.

Stevens, Stanley. 1937. "A Scale for the Measurement of the Psychological Magnitude Pitch." *The Journal of the Acoustical Society of America* 8 (3): 185.

Stevens, Stanley, John Volkmann, and Edwin Newman. 1937. "A Scale for the Measurement of the Psychological Magnitude Pitch." *J. Acoust. Soc. Amer.* 19: 14–19.

Terasawa, Hiroko, Malcolm Slaney, and Jonathan Berger. 2005. "The Thirteen Colors of Timbre." In *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics, 2005.*, 323–326.

Tversky, Amos. 1977. "Features of Similarity." *Psychological Review* 84 (4): 327–352.

Tzanetakis, George, and Perry Cook. 2002. "Musical Genre Classification of Audio Signals." *IEEE Transactions on Speech and Audio Processing* 10 (5): 293–302.

Tzanetakis, George, Georg Essl, and Perry Cook. 2001. "Automatic Musical Genre Classification Of Audio Signals." In *Proceedings of ISMIR 2001: The International Conference on Music Information Retrieval and Related Activities*, Indiana, USA.

Vapnik, Vladimir. 1999. *The Nature of Statistical Learning Theory (Information Science and Statistics)*. 2nd ed. Springer.

Wang, Fei, Xin Wang, Bo Shao, and Tao Li. 2009. "Tag Integrated Multi-label Music Style Classification with Hypergraph." In *Proceedings of 10th International Society for Music Information Retrieval Conference*, 363–368.

Warner, Timothy. 2003. *Pop Music : Technology and Creativity : Trevor Horn and the Digital Revolution*. Aldershot: Ashgate.

Weinstein, Deena. 2000. *Heavy Metal : The Music and Its Culture*. Boulder, Colorado: Da Capo Press.

Wessel, David. 1975. "Timbre Space as a Musical Control Structure." *Computer Music Journal* 3 (2): 45–52.

West, Kris. 2008. "Novel Techniques for Audio Music Classification and Search." *Doctoral Dissertation*. University of East Anglia.

Wu, Jixia, and Wenwen Cao. 2011. "Ma Yinchu's Personality and Its Formation and Development: An Application of Multidimensional Scale Analysis." In *2011 International Conference on Control, Automation and Systems Engineering (CASE)*, 1–4.

Yang, Yi-hsuan, Chia-chu Liu, and Homer Chen. 2006. "Music Emotion Classification." In *Proceedings of the 14th Annual ACM International Conference on Multimedia - MULTIMEDIA '06*, 81–84. New York, New York, USA: ACM Press.