DISSERTATION

Computational Models of Music Similarity and their Application in Music Information Retrieval

ausgeführt zum Zwecke der
Erlangung des akademischen Grades
Doktor der technischen Wissenschaften

unter der Leitung von

Univ. Prof. Dipl.-Ing. Dr. techn. Gerhard Widmer
Institut für Computational Perception
Johannes Kepler Universität Linz

eingereicht an der

Technischen Universität Wien
Fakultät für Informatik

von

Elias Pampalk
Matrikelnummer: 9625173
Eglseegasse 10A, 1120 Wien

Wien, März 2006

Kurzfassung

Die Zielsetzung dieser Dissertation ist die Entwicklung von Methoden zur Unterstützung von Anwendern beim Zugriff auf und bei der Entdeckung von Musik. Der Hauptteil besteht aus zwei Kapiteln.

Kapitel 2 gibt eine Einführung in berechenbare Modelle von Musikähnlichkeit. Zudem wird die Optimierung der Kombination verschiedener Ansätze beschrieben und die größte bisher publizierte Evaluierung von Musikähnlichkeitsmassen präsentiert. Die beste Kombination schneidet in den meisten Evaluierungskategorien signifikant besser ab als die Ausgangsmethode. Besondere Vorkehrungen wurden getroffen um Overfitting zu vermeiden. Um eine Gegenprobe zu den Ergebnissen der Genreklassifikation-basierten Evaluierung zu machen, wurde ein Hörtest durchgeführt. Die Ergebnisse vom Test bestätigen, dass Genre-basierte Evaluierungen angemessen sind um effizient große Parameterräume zu evaluieren. Kapitel 2 endet mit Empfehlungen bezüglich der Verwendung von Ähnlichkeitsmaßen.

Kapitel 3 beschreibt drei Anwendungen von solchen Ähnlichkeitsmassen. Die erste Anwendung demonstriert wie Musiksammlungen organisiert and visualisiert werden können, so dass die Anwender den Ähnlichkeitsaspekt, der sie interessiert, kontrollieren können. Die zweite Anwendung demonstriert, wie auf der Künstlerebene Musiksammlungen hierarchisch in sich überlappende Gruppen organisiert werden können. Diese Gruppen werden mittels Wörter von Webseiten zusammengefasst, welche mit den Künstlern assoziiert sind. Die dritte Anwendung demonstriert, wie mit minimalen Anwendereingaben Playlisten generiert werden können.

Abstract

This thesis aims at developing techniques which support users in accessing and discovering music. The main part consists of two chapters.

Chapter 2 gives an introduction to computational models of music similarity. The combination of different approaches is optimized and the largest evaluation of music similarity measures published to date is presented. The best combination performs significantly better than the baseline approach in most of the evaluation categories. A particular effort is made to avoid overfitting. To cross-check the results from the evaluation based on genre classification a listening test is conducted. The test confirms that genre-based evaluations are suitable to efficiently evaluate large parameter spaces. Chapter 2 ends with recommendations on the use of similarity measures.

Chapter 3 describes three applications of such similarity measures. The first application demonstrates how music collections can be organized and visualized so that users can control the aspect of similarity they are interested in. The second application demonstrates how music collections can be organized hierarchically into overlapping groups at the artist level. These groups are summarized using words from web pages associated with the respective artists. The third application demonstrates how playlists can be generated which require minimum user input.

Acknowledgments

This work was carried out while I was working as a researcher at the Austrian Research Institute for Artificial Intelligence (OFAI). I want to thank *Gerhard Widmer* for giving me the opportunity to work at OFAI and supervising me. His constant feedback and guidance have shaped this thesis from the wording of the title to the structure of the conclusions. I want to thank *Andreas Rauber* for reviewing this thesis, for convincing me to start working on MIR in the first place (back in 2001), and for his supervision during my Master's studies.

Colleagues at OFAI

I want to thank my colleagues at OFAI for lots of interesting discussions, various help, and making life at OFAI so enjoyable: *Simon Dixon*, *Werner Goebl*, *Dominik Schnitzer*, *Martin Gasser*, *Søren Tjagvad Madsen*, *Asmir Tobudic*, *Paolo Petta*, *Markus Mottl*, *Fabien Gouyon*, and those who have moved to Linz in the meantime: *Tim Pohle*, *Peter Knees*, and *Markus Schedl*. I especially want to thank: *Simon* for pointers to related work, help with signal processing, Linux, English expressions, lots of useful advice, and helpful comments on this thesis; *Werner* for introducing me to the fascinating world of expressive piano performances; *Dominik* for some very inspiring discussions and for the collaboration on industry projects related to this thesis; *Martin* for helping me implement the "Simple Playlist Generator" and for lots of help with Java; *Paolo* for always being around when I was looking for someone to talk to; *Arthur* for interesting discussions related to statistical evaluations; *Markus M.* for help with numerical problems, *Fabien* for interesting discussions related to rhythm features and helpful comments on this thesis; *Tim* for the collaboration on feature extraction and playlist generation; *Peter* for the collaboration on web-based artist similarity; and *Markus S.* for the collaboration on visualizations. I also want to thank the colleagues from the other groups at OFAI, the system administrators, and the always helpful secretariat. It was a great time and I would have never managed to complete this thesis without all the help I got.

Other Colleagues

I want to thank *Xavier Serra* and *Mark Sandler* for inviting me to visit their labs where I had the chance to get to know many great colleagues. I want to thank *Perfecto Herrera* for all his help during my visit to the Music Technology Group at UPF, for his help designing and conducting listening tests (to evaluate similarity measures for drum sounds), and for many valuable pointers to related work. I'm also very grateful to Perfe for his tremendous efforts writing and organizing the SIMAC project proposal. Without SIMAC this thesis might not have been related to MIR. (Special thanks to the European Commission for funding it!) *Juan Bello* has been very helpful in organizing my visit to the Centre for Digital Music at QMUL. I'm thankful for the interesting discussions on chord progressions and harmony in general. Which brings me to *Christopher Harte* who has taught me virtually everything I know about chords, harmony, and tonality. I'm also grateful for the guitar lessons he gave me. I want to thank *Peter Hlavac* for very interesting discussions on the music industry, and in particular for help and ideas related to the organization of drum sample libraries. I want to thank *John Madsen* for insightful discussions which guided the direction of my research. I'm grateful to many other colleagues in the *MIR community* who have been very helpful in many ways. For example, my opinions and views on evaluation procedures for music similarity measures were influenced by the numerous discussions on the MIREX and Music-IR mailing lists. In this context I particularly want to thank *Kris West* for all his enthusiasm and efforts. I'm also thankful to: *Jean-Julien Aucouturier* for a helpful review of part of my work; *Beth Logan* for help understanding the KL-Divergence; *Masataka Goto* for giving me the motivation I needed to finish this thesis a lot sooner than I would have otherwise; *Stephan Baumann* for interesting discussions about the future of MIR; and *Emilia Gómez* for interesting discussions related to harmony. Special thanks also to *Markus Frühwirth* who helped me get started with MIR.

Friends & Family

I'm very grateful to my friends for being who they are, and for helping me balance my life and giving me valuable feedback and ideas related to MIR applications. I especially want to thank the Nordsee gang for many very interesting discussions. Finally, I want to thank those who are most important: my parents and my two wonderful sisters. I want to thank them for sharing so much with me, for always being there for me, for all the love, encouragement, and support.

Financial Support

This work was supported by the EU project SIMAC (FP6-507142) and by the project Y99-INF, sponsored by the FWF in the form of a START Research Prize.

Contents

Chapter 1
Introduction

This chapter briefly describes the motivation and context of this thesis. In Section 1.1 the thesis is outlined (including a summarization of the major contributions). In Section 1.2 some examples are given which explain the syntax of Matlab expressions. (In Chapter 2 Matlab syntax is used to define algorithms.)

Motivation

The value of a large music collection is limited by how efficiently a user can explore it. Portable audio players can store over 20,000 songs and online music shops offer more than 1 million tracks. Furthermore, a number of new services are emerging which give users nearly unlimited access to music (see e.g. [KL05]).

New tools are necessary to deal with this abundance of music. Of particular interest are tools which can give recommendations, create playlists, and organize music collections. One solution is to utilize the power of Internet communities with techniques such as collaborative filtering. Furthermore, communities can share playlists as well as lists of their favorite artists.

This thesis deals with tools which do not use input from communities. The first part of this thesis is on computational models of audio-based music similarity. The second part demonstrates applications which can be built on such similarity measures.

Music Information Retrieval (MIR)

Music Information Retrieval is an interdisciplinary research field which deals with techniques to search and retrieve music related data. A list of relevant topics can be found online.[1] The topics include, among many others, computational models of music similarity and their applications.

The major MIR conference is the annual ISMIR International Conference on Music Information Retrieval which started in the year 2000.[2] Many of the papers referenced in this thesis have been published at ISMIR conferences. Besides the conferences the primary communication channel within the community is the MUSIC-IR mailing list.[2]

[1] http://ismir2006.ismir.net/callforpapers.html
[2] http://www.ismir.net

Related Music Services

The overall goal of this thesis is to support users in accessing and discovering music. There are a number of music services available which already provide this functionality. Particularly well known are Apple's iPod and the associated iTunes Music Store which offers its customers various ways of discovering music. Recently the billionth song was sold over the portal. Apple's iTunes, Amazon, and other online stores are changing the distribution channels for music. Brick and mortar stores cannot compete with unlimited shelf space, highly efficient recommendations based on customer profiles, and 24/7 opening hours (every day a year).

Furthermore, a number of services offer (nearly or completely free) access to huge music collections. Such services often have the form of easily personalizable Internet radio stations and are powerful tools to discover or simply enjoy new music. Such services include Yahoo! Launchcast or Musicmatch, Pandora, and Last.FM.

There are numerous other services which supply metadata (and automatically tag music), organize personal collections, create playlists, or support searching similar artists. These services, tools, or projects include FreeDB, MusicMoz, MusicBrainz, MusicMagic Mixer by Predixis, GraceNote's recommendation engine, Musiclens, Liveplasma, and Gnoosic. Of particular interest are Internet platforms which allow communities to exchange their music experiences. Such platforms include UpTo11, LiveJournal, Audioscrobbler, Webjay, and MySpace.

All in all, tools which enhance the experience of listening to music are rapidly developing. In this context the content-based similarity measures and the applications described in this thesis are a small building block to improve these tools.

1.1 Outline of this Thesis

This thesis consists of two major chapters. Chapter 2 deals with audio-based similarity measures. First, an introduction to similarity measures is given and state-of-the-art techniques are described. Second, combinations of these techniques are optimized and evaluated. The presented evaluation is the largest published to date. Six different collections with a total of over 20'000 pieces from over 60 genres are used. Various precautions are taken to avoid overfitting. The appropriateness of using genre classification based evaluations to explore large parameter spaces is confirmed by a listening test. In particular, the results of the listening test show that the differences measured through genre-based evaluations correspond to human listeners' ratings.

One of the findings of Chapter 2 is that *overfitting* is a very critical issue. Overfitting occurs when the performance of a similarity measure is so highly optimized for a specific music collection that its performance on a different collection is poorer than the non-optimized version. The demonstrated precautions in this thesis are highly recommendable for further work in this direction. In particular, (when using genre classification based evaluations) these are:

1. Artist filter
   An artist filter should be used to avoid overestimating the performance of the algorithm and measure improvements more accurately. An artist filter ensures that the test and training set contain different artists. As shown in this thesis, the classification accuracies are significantly lower if such a filter is used.

2. Multiple collections
   Several collections should be used both for training and for testing. As shown in this thesis, there is a high risk of overestimating performances when generalizing results based solely on one collection. The collections should be from different sources. (They should not only have different audio files but also have different genre taxonomies.)

3. Different methods to measure the same similarity aspect
   In Chapter 2 different similarity measures are combined. The goal is to combine aspects of similarity which complement each other. Given such a combination one measure can be replaced by another if it describes the same aspect. The resulting non-systematic deviations indicate how robust the combination is and how well the results can be generalized.

A further outcome of Chapter 2 are recommendations on the use of audio-based similarity measures. For a standard application the combination which performs best in the experiments described in this thesis is recommended. Compared to the baseline approach this combination is more robust towards anomalies in the similarity space (e.g. the triangular inequality is fulfilled more often), it performs best on all collections (although on the largest collection the improvements are not significant), and it is computationally not significantly more expensive than the baseline.

Chapter 3 describes three applications of similarity measures. For each application the necessary techniques (e.g. Aligned Self-Organizing Maps) are described, a demonstration is given, and limitations are discussed. The applications are:

1. Islands of Music
   A metaphor of geographic maps is used to visualize the structure of a music collection. Islands represent groups of similar pieces. Similar pieces are located close to each other on the map. A technique which allows the user to gradually shift the focus between different aspects of similarity is described. When the focus is shifted the islands are gradually rearranged according to the changing focus on similarity aspects.

2. Fuzzy Hierarchical Organization Music collections are hierarchically organized into overlapping groups. This is done at the artist level. That is, instead of pieces of music (as in the other applications) the smallest entity are artists. Each group of similar artists is summarized with words co-occurring on web pages containing the artists' names.

3. Dynamic playlist generation
   The user interaction necessary to generate a playlist is minimized. Using the skip button, the users interactively teach the system their current listening preferences. An evaluation using hypothetical usage scenarios shows that the suggested heuristic drastically reduces the number of necessary skips.

1.1.1 Contributions of this Doctoral Thesis

The contributions can be divided into two categories. The contributions described in Chapter 2 deal with (mainly audio-based) music similarity measures. The contributions in Chapter 3 deal with the application of similarity measures. Most of the work was carried out in close collaboration with a number of coauthors.

Similarity Measures

- Development of evaluation procedures and evaluation of similarity measures [PDW03b; Pam04; PFW05b] including the procedures and evaluation presented in this thesis, which is the largest evaluation published to date. Important findings include the necessity of using an artist filter, and the fact that genre classification based evaluations can be used instead of listening tests to efficiently evaluate large parameter spaces.

- Development of the freely (GNU/GPL) available Matlab MA Toolbox [Pam04][3] which implements several audio similarity measures, among them the approach presented by Aucouturier & Pachet [AP02a] which is described in this thesis (G30). Furthermore, all necessary code to implement G1C, the similarity measure which performs best in the evaluations presented in Chapter 2, is listed in this thesis.

- Development of new features (e.g. Gravity and Focus [PFW05b]), modification of similarity measures (e.g. the MFCC-based version of the Fluctuation Patterns and the G30S approach described in this thesis and in [Pam05]), and the development of optimized combinations [PFW05b] (including the new combination presented in this thesis).

Applications

- Development of the Aligned Self-Organizing Map algorithm and demonstration of its ability to organize and visualize music collections [PGW03; PDW03a; Pam03; PDW04]. In addition, this algorithm has been successfully applied to analyze expressive piano performances [PGW03; GPW04]

- Development and demonstration of an approach to hierarchically cluster music collections into overlapping groups and summarize groups of similar artists with words [PFW05a]. This clustering technique was presented in [PHH04] (it was developed to organize large drum sample libraries) and is based on insights gained in work presented in [PWC04].

- Development of a simple approach to dynamically generate playlists using minimal user interaction based on skipping behavior [PPW05a]. (Including an effective evaluation procedure to evaluate the suggested heuristics.)

1.2 Matlab Syntax

In Chapter 2 all code necessary to implement the similarity measure which performs best in the evaluations is given. Matlab syntax was chosen because

[3] http://www.ofai.at/~elias.pampalk/ma

of its compact form, and because numerous good tutorials are available online. Furthermore, all code in this thesis should be compatible with Octave[4] which is freely (GNU/GPL) available.

This section briefly describes the syntax used in the next chapter. Matlab has a similar expression syntax to standard programming languages. There are several data types in Matlab. The most important is the standard type which is a 2-dimensional matrix. In general a matrix is n-dimensional. The data type of the values is automatically determined and can be, e.g., double (by default), single, logical, complex, etc. In contrast to most other programming languages the index of the first element in an array in Matlab is always 1 (not 0).

In the following code fragment the first line creates a matrix `A` with 6 elements. All other lines link the expression on the left hand side of the equation to the right hand side (all equations are true). The intention is to explain the left hand side. The following lines of code are examples of how a matrix can be indexed:

```
A = [11 12; 21 22; 31 32]; %% 3 rows, 2 columns
size(A) == [3 2]     %% size(A,1) == 3, size(A,2) == 2
length(A) == 3       %% length(A) == max(size(A))
A(1,2) == 12         %% 1st row, 2nd column
A(1,:) == [11 12]    %% index all columns using ":"
length(A(:)) == 6    %% a 2-dim matrix can be interpreted as vector
                     %% if only one instead of two indices are used
A(end-3) == 31       %% "end" in this case equals 6
```

The "%%" is used to mark comments. Sometimes "..." is used to continue long lines in the next line. The following operators are frequently used: assignment "=", equal "==", and not equal "~=". Here are some additional examples for Matlab syntax:

```
1:2:5 == [1 3 5]   %% vector: "start value : step size: end value"
1:10  == 1:1:10
A(1:2:3,1) == [11; 31] %% result is a column vector
[11; 31]' == [11 31]   %% quote transposes matrix/vector
A(1,1:end) == A(1,:)   %% in this case: "1:end == 1:3"
A([3 2 1],[2 2]) == [32 32; 22 22; 12 12]
A(A~=31 & A>20)' == [21 22 32]
(A(:)' == 31) == [0 0 1 0 0 0]
find(A==31) == 3                 %% index of an element
linspace(a,b,n) == a:(b-a)/(n-1):b %% linearly spaced with n steps
```

[4]http://www.octave.org

A frequently used operator is the (matrix) multiplication "*" and the element-wise multiplication ".*". The following code demonstrates the use of loops and a matrix multiplication of A with B.

```
%% C = A*B;
%% size(A,2) == size(B,1)
C = zeros(size(A,1),size(B,2)); %% allocate memory
for i = 1:size(A,1),
   for j = 1:size(A,2),
      for k = 1:size(B,2),
         C(i,k) = C(i,k) + A(i,j)*B(j,k);
      end
   end
end
```

Without memory allocation (creating a matrix of zeros) the code is still correct, but would run much slower as Matlab would create a new (bigger) matrix in every loop of the iteration and copy the values of the old matrix.

Finally, here is the code used to load the 2 minutes from the center of a WAV file into Matlab. The amplitude is normalized for the Decibel computations (see Algorithm 2.7 on page 21).

```
function wav = loadwav(filename)
    max_dB = 96;
    [siz fs] = wavread(filename,'size'); %% use max (center) 2min
    if fs~=22050,
        error([filename,': fs~=22050 (',num2str(fs),')'])
    end
    if siz(1) > fs*120,
        x0 = ceil(siz(1)/2 - fs*60);
        x1 = floor(siz(1)/2 + fs*60);
    else
        x0 = 1;
        x1 = siz(1);
    end
    %% wavread returns values in the range -1 to 1
    wav = wavread(filename,[x0 x1]) * (10^(max_dB/20));
%% end loadwav
```

Chapter 2
Audio-based Similarity Measures

Computational models of audio-based music similarity are the core component of two of the applications described in Chapter 3. This chapter gives an introduction to such similarity measures. Different approaches, which complement each other, are described.

The combination of these is optimized and a large scale evaluation is conducted (including a listening test). Overall, the chosen optimization and evaluation procedures, including different approaches to avoid overfitting (such as using an artist filter), can serve as a guideline for future work in this direction. The best combination outperforms the baseline (which is the best individual similarity measure) on all of the collections used for the experiments, and is significantly more robust with respect to anomalies in the similarity space. However, on the largest test collection the performance improvements are negligible.

This chapter concludes with a recommendation for the computation of similarity which considers computation time, robustness, and quality. All details necessary (i.e. Matlab code) to implement this recommendation are given in this chapter.

2.1 Introduction

The perception of music similarity is subjective and context dependent. Relevant aspects of similarity include: instrumentation, timbre, melody, harmony, rhythm, tempo, mood, lyrics, sociocultural background, structure, and complexity. The targeted application defines the type of similarity which is of interest. For example, is the similarity measure intended to support musicologists analyzing pieces with a similar structure in a large archive? Or is it used by listeners who love classical music to explore similar interpretations of their favorite sonatas? Or is it used to replace the random shuffle function of a mobile audio player? (Such a playlist generation algorithm is one of the intended applications of the similarity measures described in this chapter.) Generally, in this thesis any definition of similarity is of interest which can be used for the applications described in Chapter 3. For example, similarity can be defined in terms of what pieces in the same playlist have (or should have) in common.

Context-dependent Similarity

The context of music similarity depends on the application and the user. In an ideal case the users could easily define the aspect of similarity they are currently interested in. However, as the low-level features used in this thesis are more or less meaningless to the users, one cannot expect them to define which of these correspond to the concepts they have in mind.

Alternatively, the users could give the system some examples defining which pieces they consider to be similar and which not, implicitly defining the context. This could be extended to some form of relevance feedback where the user could interactively respond to the system. However, a relatively large number of examples are required to avoid overfitting. Furthermore, the users might get frustrated, as it is impossible for the system to understand music the same way the users do.

In particular, based on the low-level features used in this thesis it is not possible to accurately describe any of the aspects of similarities mentioned above. Thus, this section solely aims at developing a rough overall similarity which, for example, can outperform the currently very popular random shuffle function to generate playlists.

Structure of this Chapter

This chapter is structured as follows. The remainder of this section discusses which sources can be used to obtain similarity information and gives some pointers to related work. Section 2.2 describes different techniques to compute audio-based similarity. The combination of these techniques are optimized and evaluated in Section 2.3. Section 2.4 discusses limitations and possible directions for future work. Section 2.5 describes an alternative using web-based community data. Finally, Section 2.6 concludes this chapter and gives recommendations on the choice of the similarity measure depending on the application.

2.1.1 Sources of Information

Obviously, the source is the music itself and the whole sociocultural background. However, this information cannot be directly interpreted by a machine which is, e.g., trying to give a user recommendations. A machine can rely on either judgments by human experts or large communities of music listeners, or content analysis algorithms.

Experts

High quality judgments require experts. However, due to relatively high costs the fraction of music that can be dealt with is very small. The spectrum of music worth annotating with information which allows similarity computations is limited to music which is expected to sell. Examples of systems using expert judgments include the All Music Guide[1] and the Music Genome Project[2].

The trained musicians working on the Music Genome Project annotate about 7000 pieces per month, according to an interview given by Tim Westergren.[3] Each piece is described using a set of 400 "genes". The genes used to describe a piece include, for example, "mild rhythmic syncopation" or "major key tonality". It takes an expert about 20-30 minutes to annotate one piece. Considering that about 40000 new albums are released each year alone in the US makes it obvious that it will never be possible to annotate all music this way.

Communities

Large communities can deal with much more music than a few paid experts. One of the techniques to gain insights from communities is collaborative filtering. Examples include the Internet radio station Last.fm[4] and Amazon[5]. Last.fm uses user feedback (such as: "I dont like this song") to create playlists of similar songs. Amazon uses its customers shopping behaviors to recommend similar items.

The basic idea is to recommend pieces which other users with a similar profile enjoy. However, this focus on the popularity also means that it is difficult for unknown pieces (e.g. from Garage bands) to surface.

Content Analysis Algorithms

Algorithms which directly analyze the music audio content are the essence of this chapter. The data is given in the form of audio signals, lyrics, and music videos. Algorithms can deal with much more music and in different ways than large communities. Their low costs allow the application to commercially not so interesting music. Such music includes, for example, music released under

[1] http://allmusic.com
[2] http://pandora.com/mgp.shtml
[3] http://thisweekintech.com/itn6 (3 Jan 2006)
[4] http://last.fm
[5] http://amazon.com

a creative commons license[6]. However, the performance of content analysis algorithms is limited and surely not comparable to an expert's rating.

2.1.2 Related Work

A topic closely related to similarity is audio-based genre classification. One of the first systems was developed by Tzanetakis and Cook [TC02]. More recent approaches include, for example [MB03; LOL03; WC04; ME05; CEK05; LR05; WC05; MST05; SZ05].

An interesting point to mention about the work on genre classification is that there has been a lot of optimism concerning the results. Higher accuracies were reported every year. Several publications report classification accuracies beyond 80%. Some authors suggest that the errors their algorithms make are comparable to the errors humans make when classifying music.

In contrast to these findings, the genre classification results which are reported in this thesis are far from the performance one could expect from human listeners. In fact, on one of the music collections used for evaluation (DB-30) the classification accuracy is below 15%. One important issue is how to measure the classification accuracies, and in particular, the use of an artist filter which is discussed later on.

There are two links between genre classification and similarity measures. One is that similarity measures can be evaluated using a genre classification scenario (and is discussed in Section 2.3). The other is that features which work well for genre classification are likely to also work well for similarity computations. An overview and evaluation of many popular features used for classification can be found in [Poh05; PPW05b]. In addition, it has been suggested that it is possible to automatically extract features [ZP04].

Closely related is also work on self-similarity. Self-similarity is usually used to analyze the structure (in particular repetitions) of a piece. Repetitions can be used, for example, to summarize a piece. Techniques used to compute the similarity between segments are also of interest for similarity measures which compare whole pieces. Related work includes, e.g., [Foo99; Got03; OH04; Jeh05]. Directly related to summarization is also segmentation. Good segmentations could help improve the performance of a genre classifier or music similarity measure (see e.g. [WC05]).

Work directly related to spectral similarity measures is reviewed within Subsection 2.2.3. Work directly related to the evaluation of audio-based similarity measures is reviewed in Subsection 2.3.1.

[6]http://creativecommons.org

2.2 Techniques

To compute similarities it is necessary to extract *features* (also known as descriptors) from the audio signal. Examples for good features are instrumentation (e.g. is a guitar present?) or mood (e.g. is it a depressing song?). However, current state of the art techniques cannot reliably identify the instruments in a piece, and are even less reliable when it comes to determining the mood of a piece.

The application context defines which features are of interest. For example, to generate better than random playlists does not necessarily require knowledge of the melody or chords but would definitely benefit from knowledge concerning the tempo or instrumentation. Extracting good features is far from easy and involves techniques and theories from signal processing, psychoacoustics, music perception, and (exploratory) data analysis in general.

Low-Level Features

Most features discussed in this section are low-level features. This means that although they are somehow related to a human listener's perception, in general it is very difficult to assign high-level terms to them such as "timbre" or "rhythm". Developing higher level features is ongoing work in the MIR research community and is briefly discussed in Subsection 2.4.

Temporal Scope of a Feature

Features can have different temporal scopes. Some features are extracted on the frame level. That is, the audio signal is chopped into frames with a fixed length. Typical lengths vary from 12 milliseconds to 12 seconds. Alternatively, a more meaningful segmentation can be used. For example, the signal can be segmented with respect to note onsets.

If a feature has a frame or segment level scope, then it can be interpreted as a multivariate time series. Comparing time series data directly is rather difficult. In particular, repetitions, slight tempo deviations and so forth need to be dealt with. Thus, in general some form of summarization of the time series data is used to describe a whole piece. This is especially difficult because pieces are often very inhomogeneous (e.g. Bohemian Rhapsody by Queen). Simply calculating the mean of the extracted features cannot describe, for example, differences between the chorus and the verse.

Closely related to the summarization is the question of how to compare the representation of one piece with another. Depending on the chosen representation this might not be trivial. For example, some of the similarity measures described in this section (which use features with a frame-level scope) use Monte Carlo sampling or the Kullback-Leibler divergence to compare pieces.

Computational Limits

In general it is not possible to model every nerve cell in the human auditory system when processing music archives with terabytes of data. Again the intended application defines the requirements. A similarity measure that runs on a mobile device will have other constraints than one which can be run in parallel on a massive server farm. Furthermore, it makes a big difference if the similarities are computed for a collection of a few hundred pieces, or for a catalog of a few million pieces. Finding the optimal trade-off between required resources (including memory and processing time) and quality might not be trivial.

Structure of this Section

This section is structured as follows. The next subsection gives a simple introduction to similarity computations using the Zero Crossing Rate as an example. Subsection 2.2.2 describes how the time domain representation of the audio signals is transformed to the frequency domain. Subsections 2.2.3–2.2.5 describe different features and how they are used to compute similarity. The main focus is on spectral similarity (which is somehow related to timbre) and Fluctuation Patterns (which are somehow related to rhythmical properties). Subsection 2.2.6 describes how the different approaches are combined linearly. Subsection 2.2.7 describes anomalies in the similarity space. In particular, the triangular inequality does not always hold, and a few pieces are estimated to be highly similar to a very large number of pieces while others are highly dissimilar to almost all other pieces.

2.2.1 The Basic Idea (ZCR Illustration)

This subsection illustrates the concept of audio-based music similarity using the Zero Crossing Rate as example. The ZCR is very simple to compute and has been applied to speech processing to distinguish voiced sections from noise. Furthermore, it has been applied to MIR tasks such as classifying percussive sounds, or genres. For example, the winning entry of the MIREX 2005 genre classification contest used the ZCR among other features.[7]

[7]The MIREX contest will be discussed in more detail in Subsection 2.3.1.1

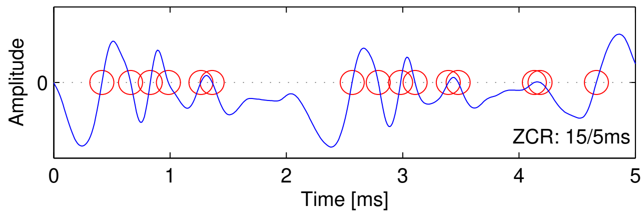


Figure 2.1: Illustration of the ZCR computation using a 5 milliseconds audio excerpt. The dotted line marks zero amplitude. The 15 circles mark the zero crossings.

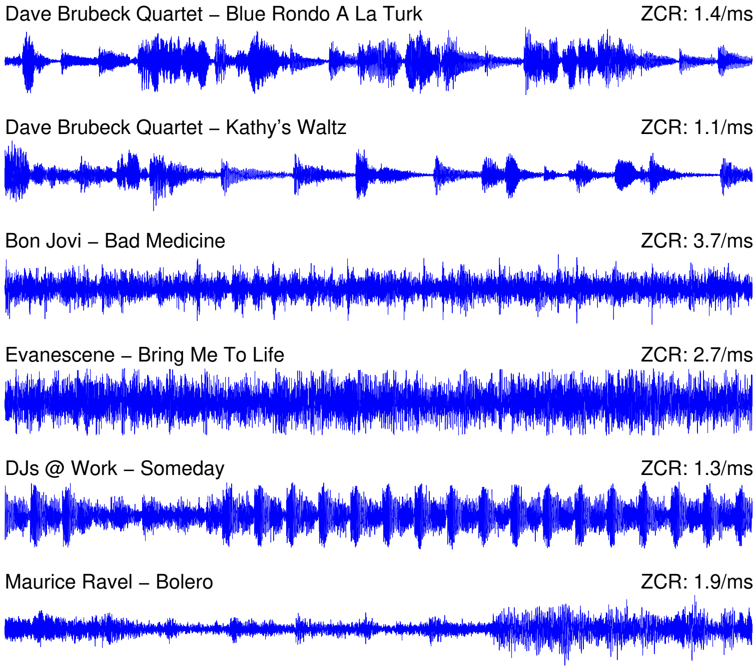


Figure 2.2: Audio excerpts with a length of 10 seconds each and corresponding ZCR values. The amplitude is plotted on the y-axis, time on the x-axis.

The ZCR is the average number of times the audio signal crosses the zero amplitude line per time unit. Figure 2.1 illustrates this. The ZCR is higher if the signal is noisier. Examples are given in Figure 2.2. The first and second excerpts (both jazz) are close to each other with values below 1.5/ms, and the third and fourth (both hard pop) are close to each other with values above 2.5/ms. Thus the ZCR seems capable of distinguishing jazz from hard pop (at least on these 4 examples).

The fifth excerpt is electronic dance music with very fast and strong percussive beats. However, the ZCR value is very low. The sixth excerpt is a soft, slow, and bright orchestra piece in which wind instruments play a dominant role. Considering that the ZCR is supposed to measure noisiness the ZCR value is surprisingly high.

The limitations of the ZCR as a noisiness feature for music are obvious when considering that with a single sine wave any ZCR value can be generated (by varying the frequency). In particular, the ZCR does not only measure noise, but also pitch. The fifth excerpt has relatively low values because its energy is mainly in the low frequency bands (bass beats). On the other hand, the sixth excerpt has a lot of energy in the higher frequency bands.

This leads to the question of how to evaluate features and to ensure that they perform well in general and not only on the subset used for testing. Obviously it is necessary to use larger test collections. However, in addition it is also helpful to understand what the features describe and how this relates to a human listener's perception (in terms of similarity). Evaluation procedures will be discussed in detail in Section 2.3.

Another question this leads to is how to extract additional information from the audio signal which might solve problems the ZCR cannot. Other features which can be extracted directly from the audio signal include, for example, the Root Mean Square (RMS) energy which is correlated with loudness (see Subsection 2.2.5). However, in general it is a standard procedure to first transform the audio signal from the time domain to the spectral domain before extracting any further features. This will be discussed in the next subsection.

2.2.2 Preprocessing (MFCCs)

The basic idea of preprocessing is to transform the raw data so that the interesting information is more easily accessible. In the case of computational models of music similarity this means drastically reducing the overall amount of data. However, this needs to be achieved without losing too much of the information which is critical to a human listener's perception of similarity.

MFCCs

Mel Frequency Cepstral Coefficients (see e.g. [RJ93]) are a standard preprocessing technique in speech processing. They were originally developed for automatic speech recognition [Opp69], and have proven to be useful for music information retrieval (see e.g. [Foo97; Log00]). Most of the features described in this section are based on MFCCs.

Alternatives include, e.g., the sonograms as used in [Pam01]. Sonograms are based on psychoacoustic models [ZF99] which are slightly more complex than those used for MFCCs. However, there are practical limitations for the complexity of the models. In particular, very accurate models, which simulate the behavior of individual nerve cells, are too computationally expensive to be applied to large music collections. Relevant Matlab toolboxes include the Auditory Toolbox[8] (which includes an MFCC implementation) and HUTear[9].

Basically, to compute MFCCs some very simple (and computationally efficient) filters and transformations are applied which roughly model some of the characteristics of the human auditory system. The important aspects of the human auditory system which MFCCs model are: (1) the non-linear frequency resolution using the Mel frequency scale, (2) the non-linear perception of loudness using decibel, and to some extent (3) spectral masking effects using a Discrete Cosine Transform (a tone is spectrally masked if it becomes inaudible by a simultaneous and louder tone with a different frequency). These steps, starting with the transformation of the audio signal to the frequency domain are described in the following paragraphs.

2.2.2.1 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 (see e.g. [Har96] or [ZF99]).

First, the signal is divided into short overlapping segments (e.g. 23ms and 50% overlap). 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 (FFT, see e.g. [PTVF92]).

The power spectrum can be computed with the following Matlab code.

[8] http://www.slaney.org/malcolm/pubs.html
[9] http://www.acoustics.hut.fi/software/HUTear

First, the variables are initialized and memory is allocated:

```
01  seg_size = 512; %% 23ms if fs == 22050Hz
02  hop_size = 256;
03  num_segments = floor((length(wav)-seg_size)/hop_size)+1;
04  P = zeros(seg_size/2+1,num_segments); %% allocate memory
05  %% hann window function, alternative: w = hann(seg_size)
06  w = 0.5*(1-cos(2*pi*(0:seg_size-1)/(seg_size-1)))';
```
(2.1)

Second, in the main loop, for each frame the FFT is computed:

```
07  for i = 1:num_segments,
08      idx = (1:seg_size)+(i-1)*hop_size;
09      x = abs(fft(wav(idx).*w)/sum(w)*2).^2;
10      P(:,i) = x(1:end/2+1);
11  end
```
(2.2)

Line 9 computes the Fourier transform (resulting in a complex vector with a length of `seg_size`). These complex values are normalized and the (real valued) power is computed. Since the Fourier transform was applied to a real valued signal, the result is symmetric and only the first half is kept for further processing in line 12. Further information on the computation of the power spectrum can be found in any standard signal processing text.

Each column in matrix `P` represents the power spectrum for a specific time frame (the number of columns is `num_segments`). Each row represents one of the `seg_size/2+1` linearly spaced frequency bins in the range of 0Hz to `fs/2`Hz (Nyquist frequency). Figure 2.3 illustrates a segment before and after applying the window function, as well as the corresponding power spectrum.

2.2.2.2 Mel Frequency

The Mel-scale [SVN37] is approximately linear for low frequencies (<500Hz), and logarithmic for higher frequencies (see Figure 2.4). 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

$$m_{\mathrm{Mel}} = 1127.01048 \log(1 + f_{\mathrm{Hz}}/700). \tag{2.3}$$

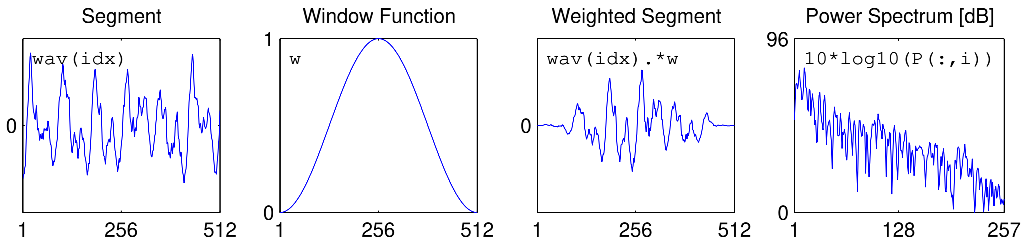


Figure 2.3: Visualization of the variables in Algorithm 2.1. Each is a one-dimensional vector. The elements of the vectors are plotted along the x-axis. The dimensions of the segments is time on the x-axis and amplitude on the y-axis. The power spectrum has the dimension Decibel (dB) on the y-axis and frequency bins on the x-axis. The first frequency bin corresponds to 0Hz, the last bin (257th in this case) to the Nyquist frequency (11025Hz).

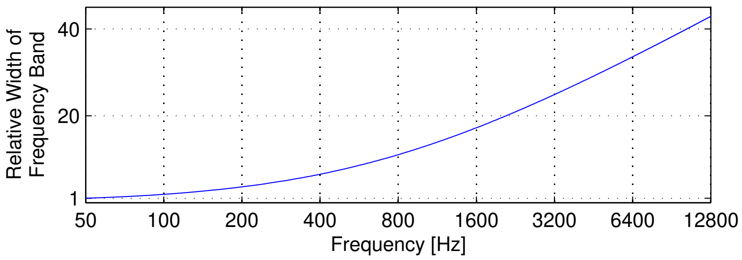


Figure 2.4: Relative width of the frequency bands according to the Mel-scale as defined in Equation 2.3. Wider frequency bands result in lower frequency resolutions. For example, the resolution at 12800Hz is over 40 times lower compared to the resolution at 50Hz.

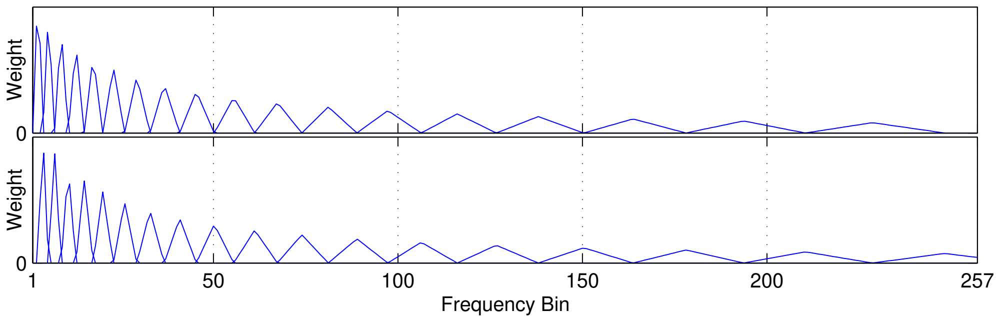


Figure 2.5: Triangular filters used to transform the power spectrum to the Mel-scale using 34 (Mel-spaced) frequency bands. The upper plot shows all uneven triangles, the lower all even ones.

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 center frequency of one triangle is the starting point for the next triangle, and the end point of the previous triangle (see Figure 2.5).

The triangular filters can be computed in Matlab as follows.[10] First, the variables are initialized and memory allocated.

(2.4)

```
num_filt = 36; %% number of Mel frequency bands

f = linspace(0,fs/2,seg_size/2+1); %% frequency bins of P
mel = log(1+f/700)*1127.01048;
mel_idx = linspace(0,mel(end),num_filt+2);
mel_filter = zeros(num_filt,seg_size/2+1);

f_idx = zeros(num_filt+2,1);
for i=1:num_filt+2,
    [tmp f_idx(i)] = min(abs(mel - mel_idx(i)));
end
freqs = f(f_idx);

%% height of triangles
h = 2./(freqs(3:num_filt+2)-freqs(1:num_filt));
```

Second, in the main loop for each triangular filter the weights are computed.

(2.5)

```
for i=1:num_filt,
    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
```

[10] This code is based on Malcolm Slaney's Auditory Toolbox.

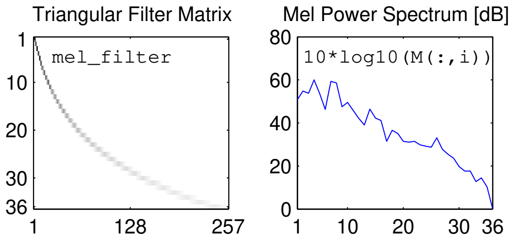


Figure 2.6: Mel filter matrix as computed in Algorithm 2.4 (high values are visualized black), and Mel power spectrum of the audio signal in Figure 2.3. The dimensions of the filter matrix are Mel frequency bands on the y-axis and FFT frequency bins (Hz) on the x-axis. The dimensions of the Mel power spectrum are dB on the y-axis and Mel on the x-axis.

In lines 18–21 the variable `freqs(i)` is the lower bound of the frequency band, `freqs(i+1)` is the center frequency, and `freqs(i+2)` is the upper bound. The `mel_filter` matrix and its effect on the power spectrum is visualized in Figure 2.6. As can be seen, the spectrum is smoothed. Specifically, details in the higher frequencies are lost. The `mel_filter` is applied by adding the following two lines to Algorithm 2.1 and 2.2.

```
04b M = zeros(num_filt,num_segments);
12b M(:,i) = mel_filter * P(:,i);
```
(2.6)

2.2.2.3 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 thesis the reference is 1, and the audio signals are rescaled with respect to this reference as described in Section 1.2. This reference is the threshold of hearing.

Decibels values are computed as follows. (First, all values smaller than the reference are set to the reference.)

```
M(M<1) = 1;
M_dB = 10*log10(M);
```
(2.7)

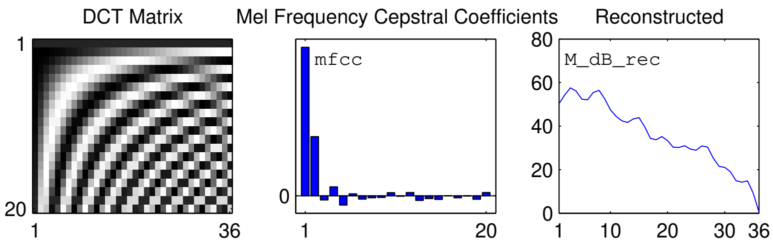


Figure 2.7: DCT matrix, MFCCs, and reconstructed Mel power spectrum (dB) for the audio signal used in Figure 2.3. High values in the DCT matrix are visualized as black. The dimensions of the DCT matrix are DCT coefficients on the y-axis and Mel frequency bands on the x-axis.

2.2.2.4 DCT

The Discrete Cosine Transform is applied to compress the Mel power spectrum. In particular, the `num_filt` (e.g. 36) frequency bands are represented by `num_coeffs` (e.g. 20) coefficients. Alternatives include, e.g., the Principial Component Analysis. 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.

The DCT matrix can be computed as follows.[11]

```
num_coeffs = 20;

DCT = 1/sqrt(num_filt/2) * ...
    cos((0:num_coeffs-1)'*(0.5:num_filt)*pi/num_filt);
DCT(1,:) = DCT(1,:)*sqrt(2)/2;
```
(2.8)

Thus the DCT matrix has `num_coeffs` rows and `num_filt` columns. Each of the rows 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 (in Decibel) as follows:

```
mfcc = DCT * M_dB;
```
(2.9)

The effects of the DCT matrix on the Mel power spectrum are shown in Figure 2.7. The resulting MFCCs are a compressed representation of the

[11] This code is based on Malcolm Slaney's Auditory Toolbox.

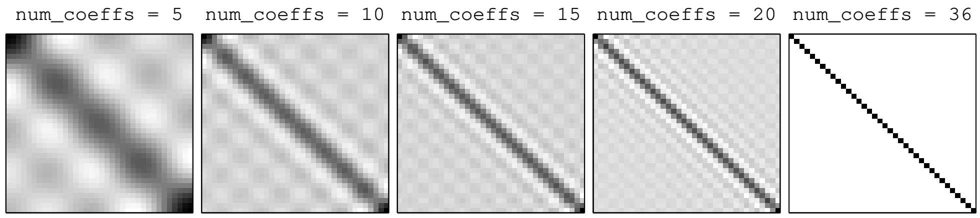


Figure 2.8: `DCT'*DCT`, each plot is a 36 by 36 matrix. The dimension of both axes are Mel frequency bands.

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 12ms (using 50% overlap for the power spectrum). Depending on the application the number of coefficients (line 1 in Algorithm 2.8) can range from 8 to 40. However, the number of coefficients is always lower than the number of Mel frequency bands used. The number of Mel frequency bands can be adjusted in Algorithm 2.4 line 1.

To understand the smoothing effects of the DCT it is useful to look at the reconstructed Mel power spectrum (Figure 2.7) and compare it to the original Mel power spectrum (Figure 2.6). The reconstructed spectrum is computed as:

```
M_dB_rec = DCT' * mfcc;
```
(2.10)

In addition, to further understand the smoothing it is useful to illustrate `DCT'*DCT` for different values of `num_coeffs` (see Figure 2.8). If the number of coefficients equals the number of Mel filters then `DCT'*DCT` is the identity matrix (thus no smoothing occurs). For lower values, the smoothing between neighboring frequency bands is clearly visible. However, it is also important to realize that the smoothing effects are not limited to neighboring frequency bands.

To conclude the computation of the MFCC coefficients Figure 2.9 illustrates the computation steps on the first 10 second sequence shown in Figure 2.2. 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).

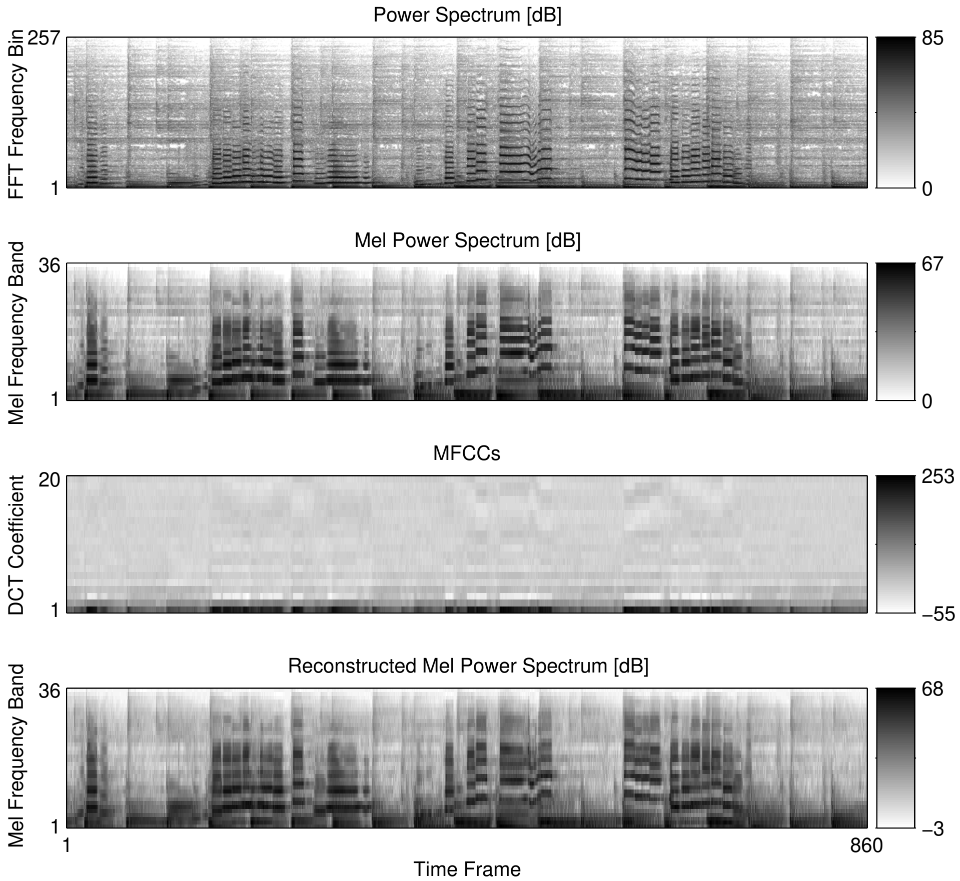


Figure 2.9: MFCC computation steps for the first audio excerpt from Figure 2.2. Time is plotted on the x-axis (the total length is 10 seconds, the temporal resolution is 256/22050 seconds which is about 12ms). The variables corresponding to each plot are `10*log10(P)`, `M_dB`, `mfcc`, and `M_dB_rec`.

2.2.2.5 Parameters

Unless stated otherwise, for the remaining subsections the following parameters are used to preprocess an audio file. First, the audio file is converted to 22050Hz mono. Second, if the piece is longer than 2 minutes, then only 2 minutes from the center are used for further processing (otherwise the whole piece is used). See Subsection 2.3.6.2 for a discussion on the effects of using only a section of the audio signal.

Third, the number of Mel filters is 36, FFT step size is 512 and hop size is 512 (no overlap, although a Hann window is used). Thus, the only difference to the parameters in Algorithms 2.1 and 2.4 is the hop size. Increasing the hop size by a factor of two also reduces computation time significantly, because the FFT is the most expensive preprocessing step. The number of DCT coefficients is 19 as recommended in [LS01]. The first coefficient which describes the average energy of each frame is ignored. In particular, the `DCT` matrix is set to `DCT=DCT(2:20,:)` before executing the code in Algorithm 2.9. As shown in [AP04a] the exact number of coefficients is not a critical parameter for spectral similarity.

2.2.3 Spectral Similarity

Spectral similarity describes aspects related to timbre. However, important timbre characteristics such as the attack or decay of a sound are not modeled. Spectral similarity is related to the timbre of a piece of music as much as color histograms are related to the color composition of a painting.

To compute spectral similarity, the audio signal is chopped into thousands of very short (e.g. 23ms) frames and their temporal order is ignored. The spectrum of each frame is described by MFCCs. The large set of frames is summarized by clustering the frames. The distance between two pieces of music is computed by comparing their cluster models.

In this subsection three approaches to compute spectral similarity are described. The main difference (by a factor of magnitudes) is the required computation time. Otherwise, all three are very similar in terms of the information they use and how they use it. However, as shown in Section 2.3 they do not behave identically and their deviations help estimate the significance of improvements.

2.2.3.1 Related Work

The first approach was presented by Foote [Foo97] based on a global set of clusters for all pieces in the collection. This global set was obtained from a classifier and is the main weak-point. To be able to describe a broad range of music a huge amount of clusters is necessary. Furthermore, there is no guarantee that music not used to train the clusters can be described meaningfully.

The first localized approach was presented by Logan and Salomon [LS01]. For each piece an individual set of clusters is used. The distances between these are computed using the Kullback-Leibler Divergence combined with the Earth Movers Distance [RTG00].

Aucouturier and Pachet suggested using the computationally more expensive Monte Carlo sampling instead [AP02a; AP04a]. A simplified approach approach using a fast approximation of the Monte Carlo sampling was presented in [Pam05]. Mandel and Ellis [ME05] propose an even simpler approach using only one cluster per piece and comparing them using the Kullback-Leibler Divergence. All three are described in detail in this subsection.

Alternative techniques to compute spectral similarity include, for example, the anchor space similarity [BEL03], the spectrum histograms [PDW03a], or simply using the mean and standard deviations of the MFCCs (e.g. [TC02]).

2.2.3.2 Thirty Gaussians and Monte Carlo Sampling (G30)

This approach was originally presented in [AP02a]. Extensive evaluation results were reported in [AP04a]. A Matlab implementation based on these won the ISMIR 2004 genre classification contest [Pam04].[12]

The approach consists of two steps. These are clustering the frames and computing the cluster model similarity. First, the various spectra (frames represented by MFCCs) which occur in the piece are summarized (i.e., the distribution is modeled) by using a clustering algorithm to find typical spectra (cluster centers) and describing how typical they are (prior probabilities), and how the other spectra vary with respect to these few typical spectra (variances).

Second, to compute the distance between two pieces, the distribution of their spectra are compared. If two pieces have similar distributions (i.e., if their spectra can be described using similar typical spectra, with similar variances, and priors) they are assumed to be similar.

Frame Clustering

The frames are clustered using a Gaussian Mixture Model (GMM) and Expectation Maximization (see e.g. [Bis95]). GMMs are a standard technique

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

used for clustering with soft assignments, or modeling probability density distributions. A reference implementation can be found, e.g., in the Netlab toolbox [Nab01] for Matlab.

A multivariate Gaussian probability density function is defined as:

$$\mathcal{N}(x|\mu,\Sigma) = \left(\frac{1}{2\pi}\right)^{p/2} |\Sigma|^{-1/2} \exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right), \tag{2.11}$$

where x is the observation (19-dimensional MFCC frame), μ is the mean (19-dimensional vector describing a typical spectrum), Σ is a 19×19 covariance matrix (for G30 only a diagonal covariance is used, i.e., only values on the diagonal are non zero).

A GMM is a mixture of M Gaussians where the contribution of the m-th component is weighted by a prior P_m with $P_m \geq 0$ and $\sum P_m = 1$:

$$p(x|\Theta) = \sum_{m=1}^{M} P_m \mathcal{N}(x|\mu_m, \Sigma_m), \tag{2.12}$$

where Θ are the parameters that need to be estimated per GMM (i.e., per piece of music): $\Theta = \{\mu_m, \Sigma_m, P_m | m = 1..M\}$. Examples of what such a model looks like for a piece of music are shown in the last paragraph of this subsection.

The optimal estimate for Θ maximizes the likelihood that the frames $X = \{x_1, ..., x_N\}$ were generated by the GMM, where N is the number of frames (which is about 5200 for the parameters used in the preprocessing). The standard measure used is the log-likelihood which is computed as:

$$L(X|\Theta) = \log \prod_n p(x_n|\Theta) = \sum_n \log p(x_n|\Theta). \tag{2.13}$$

To find good estimates for Θ a standard approach is to use the Expectation Maximization (EM) algorithm. The EM algorithm is iterative (based on an old estimate a better estimate is computed) and converges relatively fast after few iterations. The initial estimates can be completely random, or can be computed by using other clustering algorithms such as k-means. (Alternatively, as suggested in [LS01] k-means can be used alone to cluster the frames.)

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

$$P(m|x_n,\Theta) = \frac{p(x_n|m,\Theta)P_m}{p(x_n)} = \frac{\mathcal{N}(x_n|\mu_m,\Sigma_m)P_m}{\sum_{m'=1}^{M}\mathcal{N}(x_n|\mu_{m'},\Sigma_{m'})P_{m'}}. \tag{2.14}$$

The maximization step is:

$$\begin{aligned}\mu_m^* &= \frac{\sum_n P(m|x_n,\Theta)x_n}{\sum_{n'} P(m|x_{n'},\Theta)} \\ \Sigma_m^* &= \frac{\sum_n P(m|x_n,\Theta)(x_n-\mu_m)(x_n-\mu_m)^T}{\sum_{n'} P(m|x_{n'},\Theta)} \\ P_m^* &= \frac{1}{N}\sum_n P(m|x_n,\Theta).\end{aligned} \tag{2.15}$$

Cluster Model Similarity

To compute the similarity of pieces A and B a sample from each GMM is drawn, X^A and X^B respectively. (It is not feasible to store and use the original MFCC frames instead, due to memory constraints when dealing with large collections.) In the remainder of this section a sample size of 2000 is used. The log-likelihood $L(X|\Theta)$ that a sample X was generated by the model Θ is computed for each piece/sample combination (Equation 2.13). Aucouturier and Pachet [AP02a] suggest computing the distance as:

$$d_{AB} = L(X^A|\Theta^A) + L(X^B|\Theta^B) - L(X^A|\Theta^B) - L(X^B|\Theta^A). \tag{2.16}$$

Note that $L(X^A|\Theta^B)$ and $L(X^B|\Theta^A)$ are generally different values. However, a symmetric similarity measure ($d_{AB} = d_{BA}$) is very desirable for most applications. Thus, both are used for d_{AB}. The self-similarity is added to normalize the results. In most cases the following statements are true: $L(X^A|\Theta^A) > L(X^A|\Theta^B)$ and $L(X^A|\Theta^A) > L(X^B|\Theta^A)$.

The distance is (slightly) different every time it is computed due to the randomness of the sampling. Furthermore, if some randomness is used to initialize the GMM, different models will be produced every time and will also lead to different distance values.

ISMIR 2004 Genre Classification Contest

An implementation of G30 using a nearest neighbor classifier with the following parameters won the ISMIR 2004 genre classification contest. Three minutes from the center of each piece (22kHz, mono) were used for analysis. The MFCCs were computed using 19 coefficients (the first is ignored). The

FFT window size was 512 with 50% overlap (hop size 256). To initialize the GMM k-means was used. The number of samples used to compute the distance was 2000. The necessary computation time exceeded by far the time constraints of the ISMIR 2005 (MIREX) competition. The implementation is available in the MA toolbox for Matlab [Pam04].

Illustrations

Figure 2.10 shows some characteristics of G30. One of the observations is that the first and last row are very similar. That is, the original frames, and the 2000 frames sampled from the GMM generate a very similar histogram. Thus, the GMM appears to be suited to represent the distribution of the data.

A further observation is that only a few of the original frames and sampled frames have a high probability. The majority has a very low probability. This can be seen in rows 4 and 5. Note that both rows would look quite different if a new GMM is trained (or a new sample is drawn from the same GMM in the case of row 5).

Row 2 shows that most centers have a rather similar shape. Row 3 shows that the variances for some pieces are larger than for others. For example, Someday has less variance than Kathy's Waltz.

Also noticeable is the typical shape of a spectrum. In higher frequency bands there is only little energy and in the lower frequency bands there is usually more variance.

2.2.3.3 Thirty Gaussians Simplified (G30S)

G30S [Pam05] is basically a computational optimization of G30 and is based on merging ideas from [LS01] and [AP02a]. As suggested in [LS01] k-means is used to cluster the MFCC frames instead of GMM-EM. In addition, two clusters are automatically merged if they are very similar. In particular, first k-means is used to find 30 clusters. If the distance between two of these is below a (manually) defined threshold they are merged and k-means is used to find 29 clusters. This is repeated until all clusters have a least a minimum distance to each other. Empty clusters (i.e. clusters which do not represent any frames) are deleted.

The maximum number of clusters per piece is 30 and the minimum is 1. The threshold is set so that most pieces have 30 clusters and only very few have less than 20. In practice it does not occur that a piece has only 1 cluster (unless it is mostly silent). This optimization can be very useful since the distance computation time depends quadratically on the number of clusters.

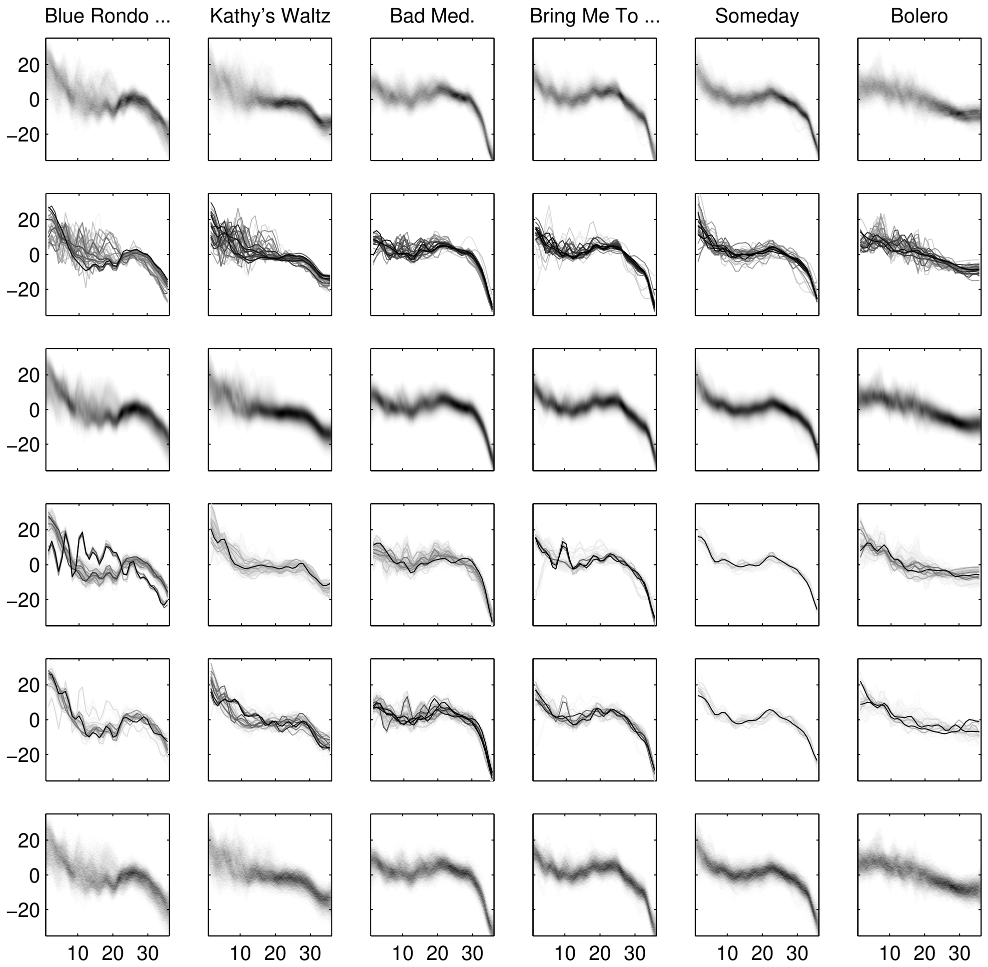


Figure 2.10: Illustration of the cluster model for G30. Each column shows different representations of one piece of music. The pieces are the same as the ones used in Figure 2.2. Each plot has the dimensions Mel frequency band on the x-axis and dB on the y-axis. That is, all MFCCs are reconstructed and visualized in the Mel space. The first row is a 2-dimensional histogram of all spectra (MFCCs). The second row are the 30 GMM centers. The gray shading corresponds to the prior of each component (black being the highest prior). The third row shows a flattened probability density of the GMM. In particular, the variance of the components are visible. The fourth row shows the probability of the original MFCC frames. The frame with the highest probability is plotted in black. The fifth row shows the probabilities of the 2000 sample frames that are drawn from the GMM. Most of them are not visible because their probability is very small compared to the highest one. The last row shows the histogram of the sample drawn.

Unlike G30 no random samples are drawn from the cluster models. Instead the cluster centers are used as sample (as suggested in [LS01]). However, instead of using the Kullback-Leibler Divergence in combination with the Earth Mover's Distance, the probability for each point of this sample is computed by interpreting the cluster model as a GMM with diagonal covariances. Since such a sample does not reflect the probability distribution (due to the different priors) the log-likelihood of each sample is weighted according to its prior before summarization:

$$L(X^A|\Theta^B) = \sum_{m=1}^{M_A} P_m^A \log\left(\sum_{m'=1}^{M_B} P_{m'}^B \mathcal{N}(\mu_m^A|\mu_{m'}^B, \Sigma_{m'}^B)\right), \qquad (2.17)$$

where M_A is the number of centers in model Θ^A. P_m^A is the prior probability of center m of model A. To compute the distances Equation 2.16 is used. $L(X^A|\Theta^A)$ can be precomputed as no new sample is drawn for each distance computation (unlike G30).

MIREX 2005 Genre Classification and Artist Identification

A nearest neighbor classifier based on a similarity measure which combined G30S with additional features was submitted to the MIREX 2005 artist identification and genre classification competitions [Pam05].[13] The main difference to the approach described here is that a FFT window length and hop size of 1024 samples was used. G30S was combined with Fluctuation Patterns and two simple descriptors (Focus and Gravitiy) which will be described later on. In terms of computation time G30S performed very well. However, in terms of the measured accuracy G30S was clearly outperformed by some of the other submissions. These results will be discussed in more detail in Section 2.3.

Illustrations

Figure 2.11 shows the same plots for G30S which were already discussed for G30 in Figure 2.10. The main observation is that G30 and G30S are highly similar. The only noticeable deviations are in rows 4 and 5. However, these rows significantly deviate if a different random sample is drawn or if the GMM is trained with a different initialization.

[13] http://www.music-ir.org/mirex2005

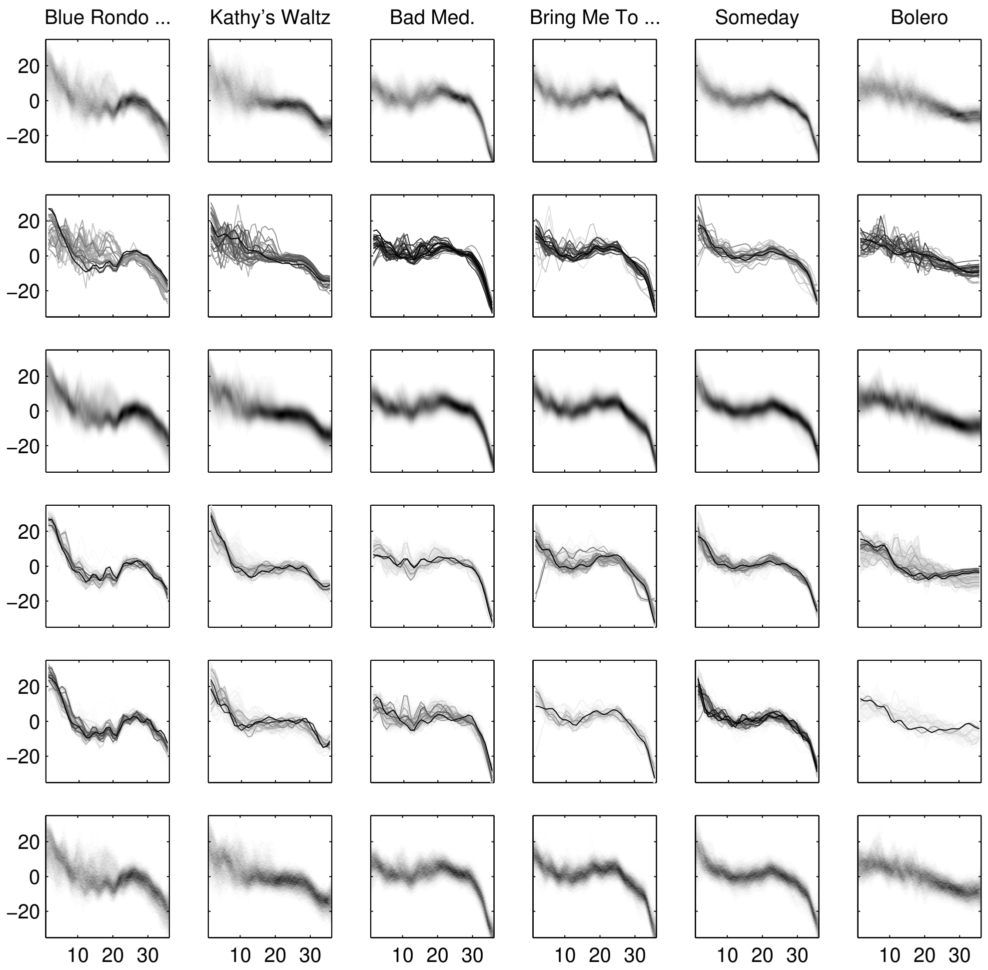


Figure 2.11: Illustration of the cluster model for G30S. The plots are generated as described in Figure 2.10.

2.2.3.4 Single Gaussian (G1)

Mandel and Ellis use a single Gaussian with full covariance to model a piece of music [ME05]. Their implementation which uses a SVM classifier won the MIREX 2005 artist identification contest and performed very well for genre classification.

To compare two Gaussians a symmetric version of the Kullback-Leibler divergence is used based on the recommendations in [MHV03]. Similar approaches are popular in the speaker identification and verification community [BMCM95].

In Matlab, the Gaussian representing a piece of music can be computed with:

```
01  m = mean(mfcc,2);  %% center                     (2.18)
02  co = cov(mfcc');   %% covariance
03  ico = inv(co);     %% inverse covariance
```

The inverse covariance is computed to avoid recomputing it for each distance computation. The symmetric Kullback-Leibler divergence between two Gaussians can be computed with:

```
04  d = trace(co1*ico2) + trace(co2*ico1) + ...      (2.19)
05      trace((ico1+ico2)*(m1-m2)*(m1-m2)');
```

The distance is rescaled to improve results when combining the spectral distance with other information. In particular the rescaled distance is computed as:

```
d = -exp(-1/fact*d).                                 (2.20)
```

The effect of rescaling is clearly visible in Figure 2.14. Using `fact=450` as rescaling factor gave best results in the combinations described later in this chapter. However, the results were stable in the range from 100 to 650. Only below 50 and above 1000 the accuracy clearly decreased.

When analyzing the effect of the rescaling on the distance matrix for a larger music collection[14] the difference can be described in terms of the ratio between the maximum value and the median value. Without rescaling, the ratio is in the order of 12 magnitudes. For `fact=10` the ratio is about 1. For `fact=5000` the ratio is about 100. For `fact=450` the ratio is about 10.

For 3 of the pieces used in the experiments reported in this thesis (i.e. 3 out of more than than 20000) G1 could not be computed. In particular, there were problems computing the inverse covariance matrix. In general

[14] The collection used is DB-L and will be described in Section 2.3.

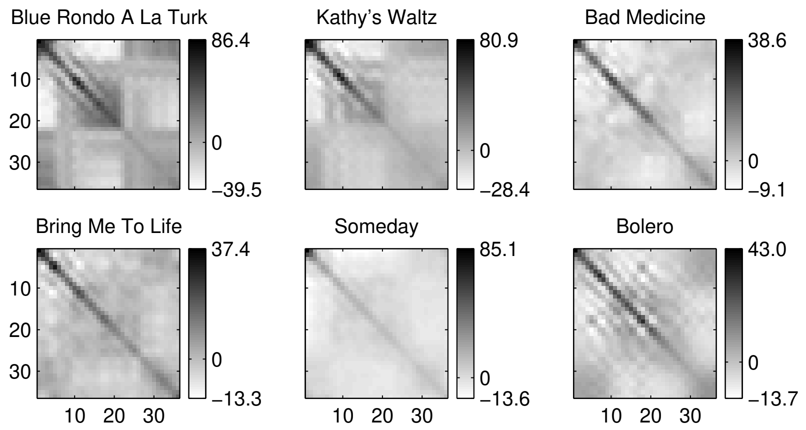


Figure 2.12: Full covariance matrices for 6 songs (G1). On both axes the dimensions are Mel frequency bands. The dimension of the gray shadings is dB.

the problem pieces had only little variance in their spectra. For example, one of them was very short (30 second) and calm. Such cases can easily be identified and excluded (e.g., all pieces can be ignored which have a value larger than 10^{10} in the inverse covariance).

Illustrations

Figure 2.12 shows the covariances for the 6 songs used in previous figures. As can be seen, there is a lot of information besides the diagonal. Noticeable are that the variances for lower frequencies are higher. Furthermore, for some of the songs there is a negative covariance between low frequencies and mid frequencies.

Figure 2.13 shows the same plots for G1 which were already discussed for G30 and G30S in Figures 2.10 and 2.11. Since G1 uses only one Gaussian, there is only one line plotted in the second row. Noticeable, is also that there are more lines visible in rows 4 and 5. This indicates there are fewer frames (sampled or original) which have much higher probabilities than all others. Otherwise, in particular the third and last row are very similar to those of G30 and G30S and indicate that the models are very similar.

2.2.3.5 Computation Times

The CPU times for G30, G30S, and G1 are given in Table 2.1. The frame clustering (FC) time is less interesting than the time needed to compute the cluster model similarity (CMS). While FC can be computed offline, either all possible distances need to be precomputed (and at least partially stored) or

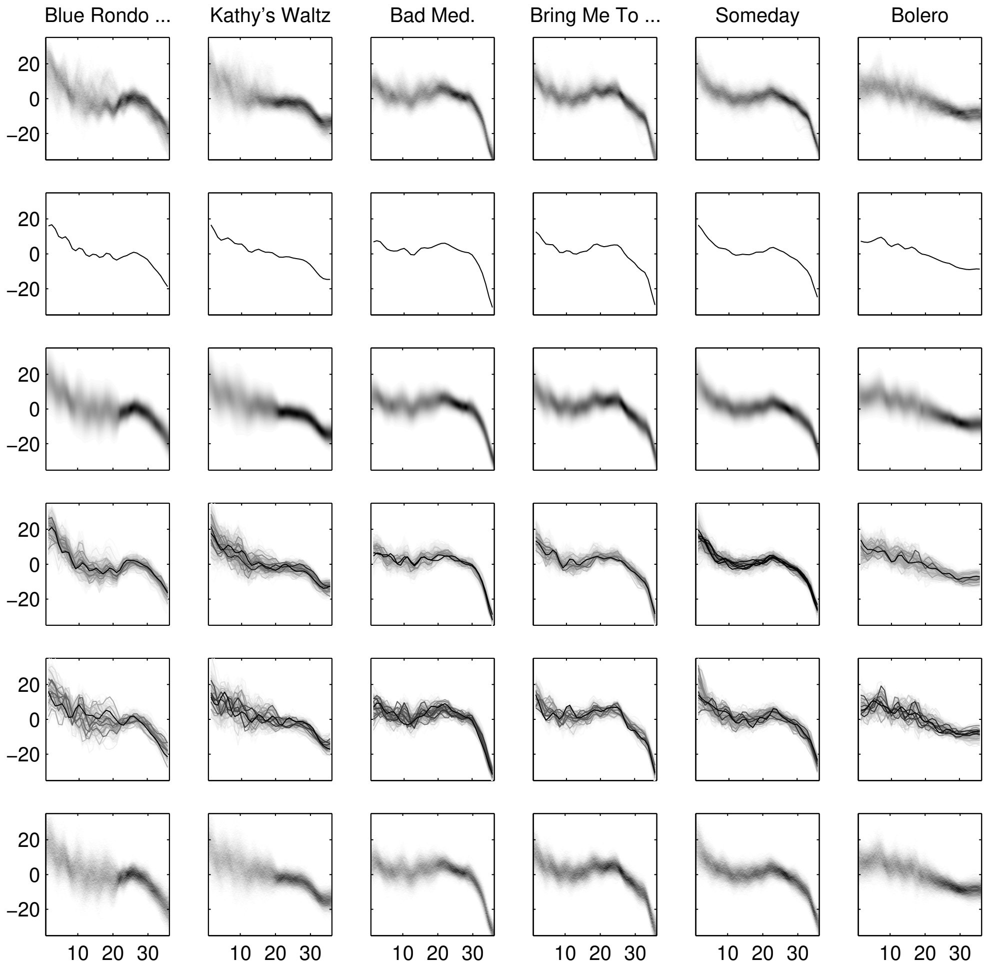


Figure 2.13: Illustration of the cluster model for G1. The plots are generated as described in Figure 2.10.

	G30	G30S	G1
FC	25000	700	30.0
CMS	400	2	0.1

Table 2.1: Approximate CPU times in milliseconds on a Intel Pentium M 2GHz (755) for frame clustering (per piece) and cluster model similarity (per pair of pieces). The approximate time for loading a 120 second (22kHz, mono) audio file in WAV format into Matlab is 0.25 seconds. The necessary time to compute MFCCs is about 1.5 seconds using no overlap between frames and a segment size of 512 (23ms). The number of frames is about 5200 frames (for 2 minutes of audio).

the system needs to compute all distances of interest very fast to minimize the system's response time. Note that the FC time for G30 can easily be reduced to the time of G30S (and is mainly a question of accuracy). However, there is no way to reduce the computation times of G30 or G30S to those of G1. G1 is clearly magnitudes faster.

2.2.3.6 Distance Matrices

Figure 2.14 shows the distance matrices for the 6 songs using the three spectral similarity measures described in this section. The matrices computed for G30 and G30s are very similar. Furthermore, the difference between the original and the rescaled distance matrix for G1 are clearly noticeable. Rescaling G1 is very important when combining the distance matrix with additional information as discussed in the subsequent sections. Furthermore, a balanced distance matrix is also important for techniques which visualize whole collections such as the Islands of Music discussed in the next chapter. However, if only a ranked list of similar pieces is required then the scaling is not critical.

Compared to the ZCR results it seems that one problem has been solved. That is, the piece of classical music is now differentiated from the other pieces. However, the hard pop and electronic dance pieces are not distinguishable. One solution is to add information related to the beats and rhythm which is the topic of the next section.

2.2.4 Fluctuation Patterns

Fluctuation Patterns (FPs) describe the amplitude modulation of the loudness per frequency band [Pam01; PRM02a] and are based on ideas developed in [Frü01; FR01]. They describe characteristics of the audio signal which are not described by the spectral similarity measure.

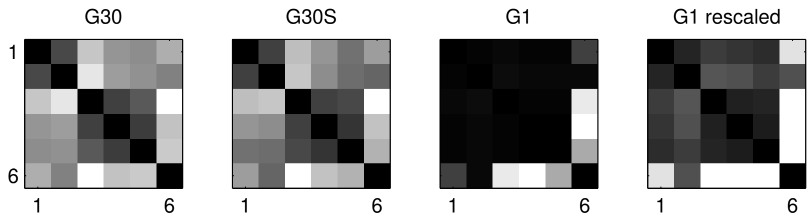


Figure 2.14: Distance matrices for the 6 songs. The songs are in the same order as in Figure 2.13. That is, the first row (and column) shows all distances to “Blue Rondo A La Turk”. The last row (and column) shows all distances to “Bolero”.

The loudness modulation has different effects on our sensation depending on the frequency. The sensation of fluctuation strength [Ter68; Fas82] is most intense around 4Hz and gradually decreases up to a modulation frequency of 15Hz (see Figure 2.15). At 15Hz the sensation of roughness starts to increase, reaches its maximum at about 70Hz, and starts to decrease at about 150Hz. Above 150Hz the sensation of hearing three separately audible tones increases [ZF99].

The following computation steps are based on a frequency/loudness/time representation. This can be a sonogram as used in [Pam01] or a Mel-frequency dB spectrogram as used in [Pam05]. An evaluation of the impact of different preprocessing steps on the accuracy for genre classification using FPs based on sonograms can be found in [LR05]. The advantage of using FPs based on MFCCs is that (when using FPs in combination with spectral similarity) the MFCCs are already available.

The 4 computation steps for FPs are: (1) Cut the spectrogram into short segments, e.g., 6 seconds as used in [Pam01] or 3 seconds as described in this subsection. (2) For each segment and each frequency band use an FFT to compute the amplitude modulation frequencies of the loudness in the range of 0-10Hz. (3) The modulation frequencies are weighted using a model of perceived fluctuation strength. (4) Some filters are applied to emphasize certain types of patterns.

The resulting FP is a matrix with rows corresponding to frequency bands and columns corresponding to modulation frequencies (in the range of 0-10Hz). The elements of this matrix describe the fluctuation strength. To summarize all FP patterns representing the different segments of a piece the median of all FPs is computed. Finally, one FP matrix represents an entire piece. The distance between pieces is computed by interpreting the FP matrix as high-dimensional vector and computing the Euclidean distance.

The following paragraphs describe the computation in more detail based on `M_dB` computed with the parameters described in Subsection 2.2.2.5. At the end of this section there are some paragraphs describing illustrations which help understand basic characteristics of the FPs.

2.2.4.1 Details

The parameters used are: segment size 128 (about 3 seconds), hop size 64 (50% overlap). Instead of the 36 frequency bands of the Mel spectrum only 12 are used. The grouping of frequency bands is described below. The resolution of the modulation frequency in the range of 0 to 10Hz is 30. This results in 360 (12×30) dimensional FPs.

The main reason for reducing the frequency resolution is to reduce the overall dimensionality of the FPs, and thus the required memory to store one pattern. Furthermore, a high frequency resolution is not necessary. For example, in [Pam01] 20 frequency bands were used. Analysis of the eigenvectors showed that especially higher frequency bands are very correlated. In [Pam05] only 12 frequency bands were used. In particular, the following mapping was used to group the 36 Mel bands into 12 frequency bands:

```
t = zeros(1,36);
t(1)   = 1; t( 7: 8) = 5; t(15:18) =  9;
t(2)   = 2; t( 9:10) = 6; t(19:23) = 10;
t(3:4) = 3; t(11:12) = 7; t(24:29) = 11;
t(5:6) = 4; t(13:14) = 8; t(30:36) = 12;

mel2 = zeros(12,size(M_dB,2));
for i=1:12,
    mel2(i,:) = sum(M_dB(t==i,:),1);
end
```

(2.21)

The actual values are more or less arbitrary. However, they are based on the observation that interesting details are often in lower frequency bands. Note that the energy is added up. Thus, the 12th frequency band of `mel2` represents the sum of 7 `M_db` bands while the first and second band only represent one. This can be seen in Figure 2.16. In particular, the frequency bands 9-11 have higher values.

The following defines the constants used for computations, in particular the fluctuation strength weights (`flux`), the filter which smoothes over the frequency bands (`filt1`), and the filter which smoothes over the modulation

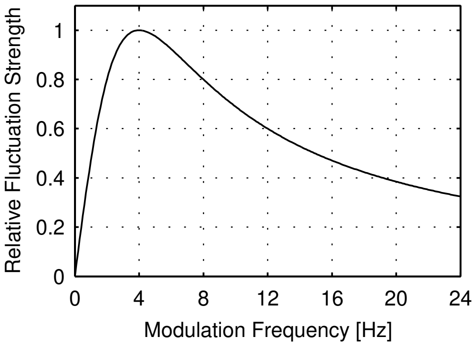


Figure 2.15: The relationship between the modulation frequency and the weighting factors of the fluctuation strength.

frequencies (`filt2`):

(2.22)

```
11  f = linspace(0,22050/512/2,64+1);
12  flux = repmat(1./(f(2:32)/4+4./f(2:32)),12,1);
13  w = [0.05, 0.1, 0.25, 0.5, 1, 0.5, 0.25, 0.1, 0.05];
14  filt1 = filter2(w,eye(12)); %% 12x12
15  filt1 = filt1./repmat(sum(filt1,2),1,12);
16  filt2 = filter2(w,eye(30)); %% 30x30
17  filt2 = (filt2./repmat(sum(filt2,2),1,30))';
```

The weighting vector `flux` for the amplitude modulation is based on a model of perceived fluctuation strength [Fas82] and was primarily designed as a model for speech. The weights are shown in Figure 2.15. The filter is applied as shown in line 25 of Algorithm 2.24.

From a practical point of view, the main purpose of the weighting is to reduce the influence of very low modulation frequencies which generally have higher energies. Alternatively, a weighting function based on preferred tempo could be used. For example, the periodicity histograms defined in [PDW03a] used a resonance model [Moe02] weighting function which has a maximum response around 120bpm (2Hz).

The two filters (`filt1` and `filt2`) are the same as used in [Pam01]. They are applied to smooth the patterns. The effect of the fluctuation strength weighting and the smoothing filters can be seen in Figure 2.16.

In general, very short pieces (e.g. shorter than 15 seconds) should be excluded from the computations. However, if for some reason there is an interest in dealing with short pieces (e.g. when computing the similarity between segments) it is necessary to pad the Mel spectrogram with zeros.

This can be done, for example, with:

```
if size(mel2,2)<128, %% pad with zeros
    mel2 = [zeros(12,ceil(128-size(mel2,2))),mel2];
end
```
(2.23)

The following algorithm shows the main part of the computation of the FP. The frequency modulation of the amplitude for each of the 12 frequency bands is computed with an FFT. Alternatively, a computationally more expensive comb-filter could be used (e.g. [PDW03a]).

The amplitude modulation is weighted according to the fluctuation strength, the difference filter is computed to emphasize vertical lines, and finally the pattern is smoothed with the filters.

```
num_segments = floor((size(mel2,2)-128)/64+1);
fp_all = zeros(num_segments,12*30);
for i=1:num_segments,
    X = fft(mel2(:,(1:128)+64*(i-1)),128,2);
    X2 = abs(X(:,2:32)).*flux; %% amplitude spectrum
    X2 = filt1*abs(diff(X2,1,2))*filt2;
    fp_all(i,:) = X2(:)';
end
```
(2.24)

For a 2 minute piece using 50% overlap and a window size of 128 (about 3 seconds) results in 79 FPs. A song can be summarized by simply computing the median of these patterns:

```
fp = median(fp_all);
```
(2.25)

Finally, to compare two pieces the Euclidean distance is computed:

```
d = sqrt(sum((fp1-fp2).^2)); %% or: d = norm(fp1-fp2);
```
(2.26)

2.2.4.2 Illustrations

Figure 2.16 shows a segment before and after the four main computation steps. The difference between mel and mel2 is clearly visible. The amplitude modulation shows the kind of information that is extracted using an FFT. For example, a vertical line in the 28th modulation frequency bin (just below 10Hz) is clearly visible. The effect of the weighting is noticeable particularly for low modulation frequencies (first and second column). One of the main

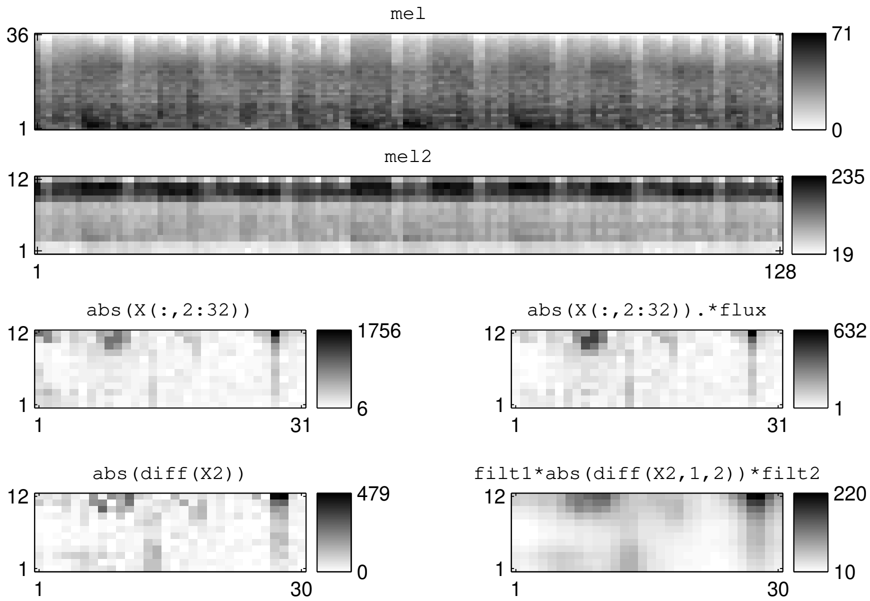


Figure 2.16: FP computation steps for one segment.

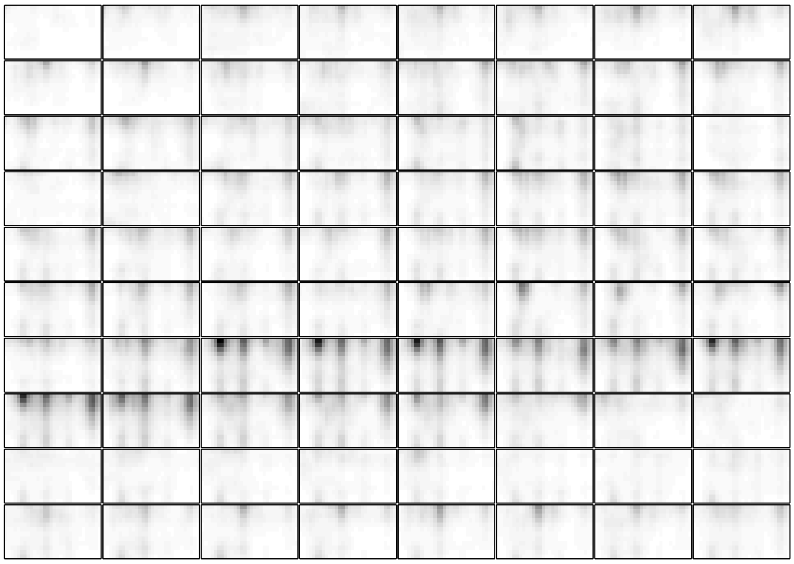


Figure 2.17: FP of all segments computed for the song Someday. The gray shadings of the plots are scaled globally. Black represents high values, white represents low values. The temporal order of the plots is left-right, top-down. The segment used in Figure 2.16 is number 15 (2nd row, 7th column). The last pattern (80th, last row, last column) is the median of all.

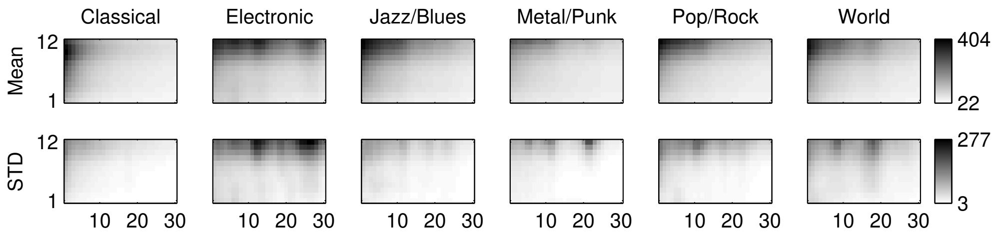


Figure 2.18: FP mean and standard deviation (STD) for different genres of the DB-MS collection.

effects of the difference filter is that the width of the vertical line in the 28th column is doubled. The effect of the smoothing filters is also clearly visible.

Figure 2.17 shows the FPs for all 79 segments computed for a song. As can be seen, the median does not reflect the characteristics of the sequence in the 7th and 8th row. In fact, the patterns of the median are hardly noticeable compared to the patterns with the highest values in this sequence. Computing the median is surely not the optimal solution. However, it is one of the simplest.

Figure 2.18 shows the differences between genres of the DB-MS collection (see Subsection 2.3.3 for details on this collection). The average patterns for classical, electronic, and punk are distinguishable. On the other hand, world, pop/rock, and jazz/blues seem rather similar.

2.2.5 Simple Features

This section describes "simple" features, that is, features which can be extracted based on the algorithms already mentioned previously with a single line (or only a few lines) of Matlab code. All of these are one-dimensional, continuous, and distances are computed as absolute difference.

These features are best characterized as low-level audio statistics. Although some higher level concepts can sometimes be associated with them, such associations might cause more confusion than help explain their true meaning. These features are intended to complement the spectral similarity and Fluctuation Patterns. The combination is described in the next section.

The features presented here are only a tiny selection of all features which have been published and used in MIR research. A recent overview of different features (also known as descriptors) can be found in [Poh05]. Slightly older but nevertheless very insightful overviews can be found in [TC02] and [AP03].

2.2.5.1 Time Domain

The main advantage of time domain features is that they can be computed very efficiently. A typical example is the ZCR already mentioned previously. In Matlab notation the average ZCR per second can be computed as (where `fs` is the sampling frequency in Hertz):

```
zcr = sum(abs(diff(sign(wav))))/2/length(wav)*fs;
```
(2.27)

Other features which can be extracted directly from the audio signal include, for example, the Root Mean Square (RMS) energy which is correlated with the perception of loudness:

```
rms = sqrt(mean(wav.^2));
```
(2.28)

2.2.5.2 Power Spectrum

Based on the power spectrum a number of features can be extracted. One example is the noisiness. Several variations exist; one option to implement it is:

```
P_dB = P;
P_dB(P_dB<1) = 1;
P_dB = 10*log10(P_dB);
P_dB_new = zeros(size(P_dB,1)-10,floor(size(P_dB,2)/10));
for j=1:floor(size(P_dB,2)/10),
    P_dB_new(:,j) = ...
       mean(P_dB(11:end,(1:10)+(j-1)*10),2);
end
noisiness = sum(sum(abs(diff(P_dB_new))));
```
(2.29)

In words this computation can be described as follows. First a smoothed version of the spectrogram (`P_db_new`) is computed as the mean energy over 10 subsequent frames (about 115ms). Frequencies below 800Hz in the spectrogram are ignored. Then for each frame in `P_db_new` the absolute difference between adjacent frequency bins is computed (`abs(diff(P_dB_new))`). Finally, the noisiness is the sum of all absolute differences. The noisiness differentiates between rather noisy or rather harmonic sounds. A noisy sound will have a low value, because the spectrum is rather flat.

On the other hand, a not so noisy signal where the sounds change within significantly less than 115ms will also be effected by this smoothing and the computed value might not correspond to the perceived noisiness. Furthermore, the noisiness as defined here is correlated with the loudness such that a loud sound will generally be less noisy. This is clearly a major limitation and could be addressed by normalizing the loudness.

2.2.5.3 Mel Power Spectrum

Several simple features can be extracted based on the reconstructed Mel power spectrum, or any other form of spectrogram (such as sonograms). Examples include loudness related features, Percussiveness (which describes the strength of percussive sounds), and the Spectral Centroid. There are different options to compute each of these descriptors; one option for each is given below.

The average loudness is an indicator of how energetic a piece is. However, the average loudness depends very much on production effects such as compression.

```
avg_loudness = mean(M_dB(:));
```
(2.30)

Computing the difference of two subsequent frames (also known as spectral difference) is a standard approach for onset detection and beat tracking (e.g. [BDA+05]). This information can also be used to compute Percussiveness. Alternatives include the approach presented in [HSG04].

```
percussiveness = mean(abs(diff(M_dB,1,2)))
```
(2.31)

The spectral centroid is correlated with the perception of brightness. In particular, sounds with more energy in higher frequencies are usually perceived as brighter. However, as a descriptor of the spectral shape the centroid is very limited and can be misleading. For example, the spectral centroid cannot appropriately describe the mixture of a very bright (high pitched) sound with a very low pitched sound.

```
spectral_centroid = ...
    sum((1:num_filt).*sum(M_dB,2)'./max(sum(M_dB(:)),eps));
```
(2.32)

2.2.5.4 Fluctuation Patterns

Based on any higher-level representation of the spectrogram it is usually possible to extract other features. The features described here have been presented in [Pam01; PFW05b]. An evaluation of a larger feature set extracted from the Fluctuation Patterns can be found in [LR05]. In the lines of code given below `FP` is defined as `FP = reshape(fp,12,30)`. That is, instead of the vector representation used in Section 2.2.4 a matrix representation is used.

The maximum fluctuation strength is the highest value in the rhythm pattern. Pieces of music which are dominated by strong beats have very high values. Typical examples with high values include electronic and house music. Whereas, for example, classical music has very low values.

```
fp_max = max(FP(:));
```
(2.33)

Alternatively, the sum can be computed:

```
fp_sum = sum(FP(:));                                              (2.34)
```

The bass is calculated as the sum of the values in the two lowest frequency bands with a modulation frequency higher than 1Hz.

```
fp_bass = sum(sum(FP(1:2,3:30)));                                 (2.35)
```

The aggressiveness is measured as the ratio of the sum of values within the frequency bands 2-12 and modulation frequencies below 1Hz compared to the sum of all. Less aggressive pieces have higher values.

```
fp_aggr = sum(sum(FP(2:end,1:4)))/max(max(FP(:)),eps);            (2.36)
```

The domination of low frequencies is calculated as the ratio between the sum of the values in the 4 highest and 3 lowest frequency bands.

```
fp_DLF = sum(sum(FP(1:3,:)))/max(sum(sum(FP(9:12,:))),eps);       (2.37)
```

Gravity

The Gravity (FP.G) describes the center of gravity of the FP on the modulation frequency axis. Given 30 modulation frequency-bins (linearly spaced in the range from 0-10Hz) the center usually lies between the 10^{th} and the 15^{th} bin, and is computed as

$$fpg = \frac{\sum_j j \sum_i \mathrm{FP}_{ij}}{\sum_{ij} \mathrm{FP}_{ij}}, \tag{2.38}$$

where i is the index of the frequency band, and j of the modulation frequency. In Matlab notation this is computed with:

```
fpg = sum(sum(FP).*(1:30))/max(sum(FP(:)),eps);                   (2.39)
```

Low values indicate that the piece might be perceived as slow. However, FP.G is not intended to model the perception of tempo. Effects such as vibrato or tremolo are also reflected in the FP.

Focus

The Focus (FP.F) describes the distribution of energy in the FP. In particular, FP.F is low if the energy is focused in small regions of the FP, and high if the energy is spread out over the whole FP. The FP.F is computed as mean value of all values in the FP matrix, after normalizing the FP such that the maximum value equals 1. The Focus is computed with:

```
fpf = mean(FP(:)./max(max(FP(:)),eps));                           (2.40)
```

2.2.5.5 Illustrations

Figure 2.19 shows the range of values per feature and genre using the DB-MS collection.[15] The overlap between genres is generally quite large. The genres are clearly distinguishable only in a few cases. For example, the average loudness feature can distinguish classical music from metal/punk. In addition, correlations are made visible, e.g., between the spectral centroid and average loudness.

Figures 2.20 and 2.21 show the correlation between distance matrices. The correlation is measured comparing the values of the upper triangle for each of the 17 distances matrices computed from the 13 features, the Fluctuation Patterns, and the 3 spectral similarity measures.

Noticeable are the high correlation of the spectral similarity measures (G30, G30S, G1), although the correlation of G1 is significantly lower. The observation that the spectral centroid is correlated with the average loudness from Figure 2.19 is confirmed. The reason for this might be that for a piece to be louder in average, the energy must be spread over many frequency bands including higher ones. As can be seen in Figure 2.19 classical pieces have a lower spectral centroid than aggressive metal/punk pieces.

Not particularly surprising is the correlation between FP Max and FP Sum, or the correlation between average loudness and RMS. However, quite surprising is the relatively high correlation between the average loudness and the spectral similarity measures. Specifically because the first coefficient (the average loudness of each frame) is not used for the spectral similarity. This indicates that not using the first coefficient does not mean that loudness information is completely removed.

Another observation is the high correlation between Percussiveness, FP, FP Max, FP Sum, and FP Bass. The reason is that the Percussiveness is high if there are strong beats. Usually beats are periodic, thus they will also have an impact on the FP. Strong beats lead to high FP Max and FP Sum values. Similar FPs will have a similar FP Sum. Furthermore, a piece that does not have very strong beats, will also not have very strong bass beats.

Also noticeable is that the correlation between ZCR and noisiness is very low. On one hand, this is due to the limitations of the noisiness descriptor discussed previously. On the other hand, the relatively high correlation between the spectral centroid and the ZCR also explains part of it.

The differences between Figures 2.20 and 2.21 show how reliable the values are. Depending on the music used, the values will vary. In this case, the deviations are around 0.1.

[15]Details of the DB-MS collection are described in Subsection 2.3.3.

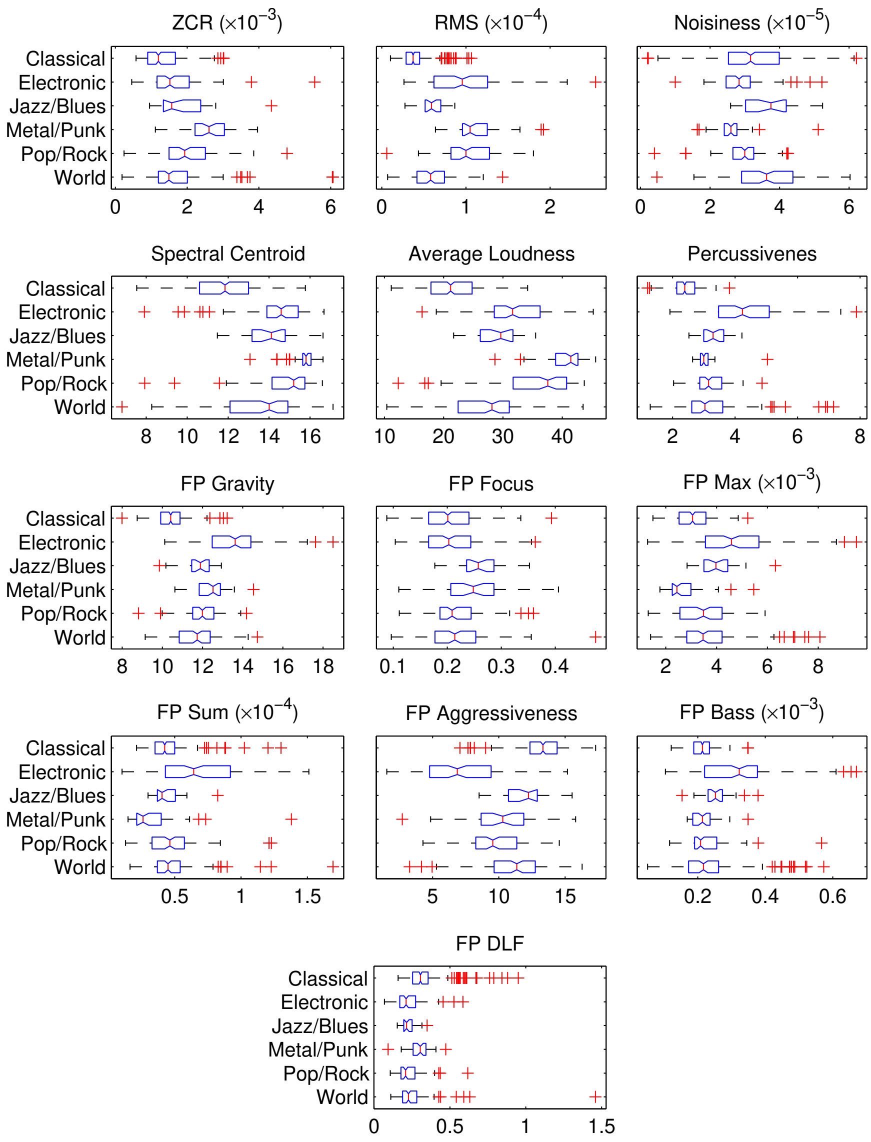


Figure 2.19: The boxes show the range of values for the 13 simple features per genre computed on the DB-MS collection (see Subsection 2.3.3). Each box has a line at the lower quartile, median, and upper quartile values. The whiskers extend from the box out to the most extreme data value within 1.5 times the interquartile range of the sample. Outliers are data with values beyond the ends of the whiskers and marked with "+". The notches represent a robust estimate of the uncertainty about the medians for box-to-box comparison. Boxes whose notches do not overlap indicate that the medians of the two groups differ at the 5% significance level (for details see the Matlab boxplot function documentation).

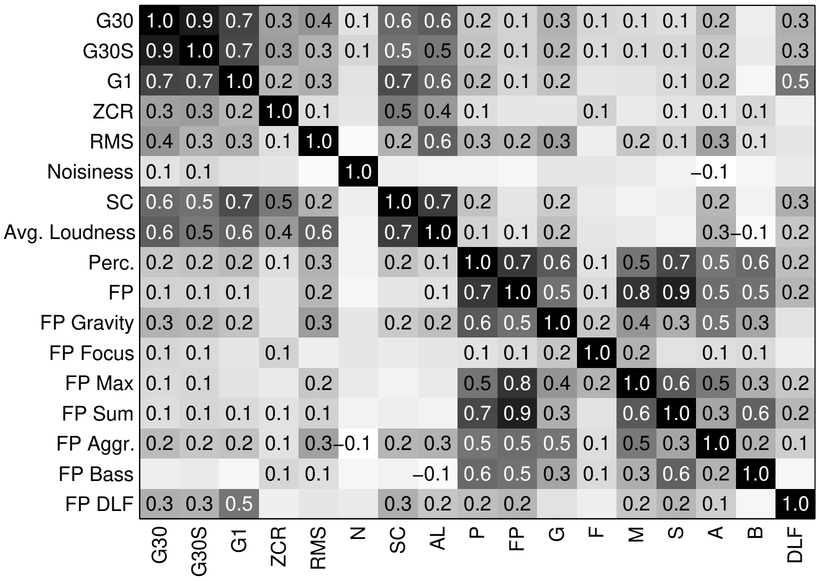


Figure 2.20: Correlation matrix for 17 distance matrices computed on the DB-MS collection (729 pieces). The values are rounded and zeros are not displayed.

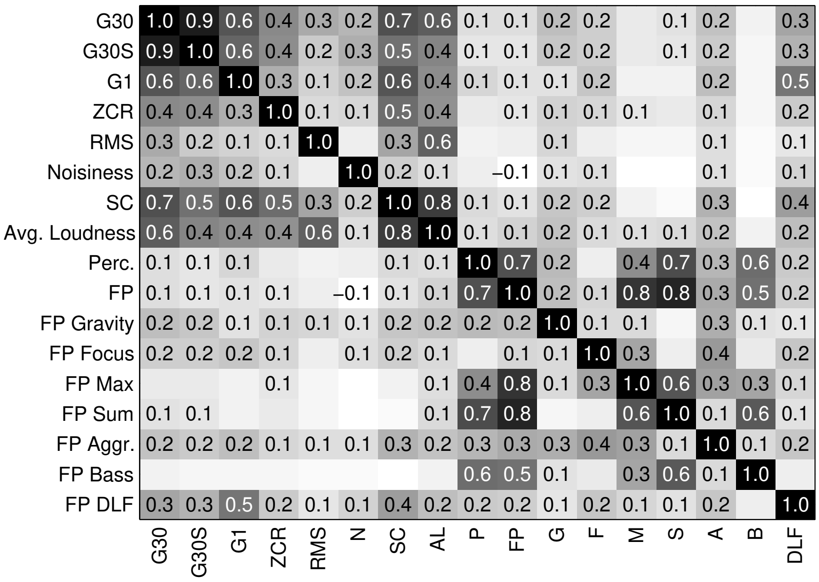


Figure 2.21: Correlation matrix for 17 distance matrices computed on the DB-L collection (2381 pieces).

2.2.6 Linear Combination

A straightforward approach to combine the spectral similarity with the Fluctuation Patterns and the simple features is to compute a weighted sum of the individual distances. This linear combination is similar to the approach used for the aligned Self-Organizing Maps (SOMs) in Section 3.2.5. Originally, the idea was that the user should be able to manually set the weights of the linear combination. However, as the meaning of the different features is not very intuitive it is necessary to determine the weights automatically. Before combining the distances, they are normalized such that the standard deviation of all pairwise distances within a music collection equals 1. For the experiments reported in Section 2.3 the mean and standard deviations were computed on the DB-L collection. In general the combination has the following form:

$$d = \sum_i w_i (d_i - \mu_i)/\sigma_i, \tag{2.41}$$

where d_i is the distance computed according to the i-th similarity measure, the normalization factor μ_i is the mean of all the distances, and σ_i is the standard deviation. The weights w_i define the contribution of each similarity measure. For example, for the MIREX'05 submission described in [Pam05] the weights were 65% for G30S, 15% for FP, 5% for FP Focus, and 15% for FP Gravity. For the best combination found in Section 2.3 (and refered to as G1C later on) the combination (including the normalization factors) is:

$$\begin{aligned} d \quad = \quad & 0.7\,(-\exp(-d_{\mathrm{G1}}/450) + 0.7950)\,/0.1493 + \\ & 0.1\,(d_{\mathrm{FP}} - 1688.4)\,/878.23 + \\ & 0.1\,(d_{\mathrm{FPG}} - 1.2745)\,/1.1245 + \\ & 0.1\,(d_{\mathrm{FPB}} - 1064.8)\,/932.79. \end{aligned} \tag{2.42}$$

There is a conceptual problem with the linear combination: A human listener does not compute a weighted sum of the similarities with respect to different aspects. In contrast, a single aspect which is similar is sufficient to consider pieces to be similar. Thus, a model which defines the overall distance to be the minimum distance of its components seems more appropriate. However, it is also necessary to realize that the aspects which are combined in this thesis are not of a kind that a human listener would ever consider when judging similarity. Nevertheless, developing alternative models to combine different similarity measures is an interesting direction for future work.

2.2.7 Anomalies

Finally, before turning to the evaluation, this subsection describes three anomalies in the similarity space. These are caused by the spectral similarity and to some extent contradict a human's intuition of a similarity space.

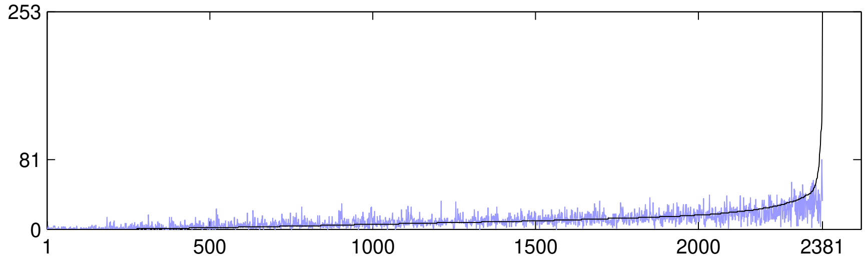


Figure 2.22: Number of times each piece in DB-L is one of the 10 closest neighbors to the other pieces. The black line is G1, the other is G1C. The pieces are sorted according to their G1 values. The highest frequency for G1 is 253, the highest for G1C is 81.

2.2.7.1 Always Similar

Aucouturier and Pachet [AP04a] pointed out that some pieces are very similar to a large number of other pieces although they are not perceptually similar (false positives).

This anomaly can be studied (as suggested in [AP04a]) by computing the 10 nearest neighbors to each piece in the collection. The examples given here are computed from the DB-L collection which has a total of 2381 tracks. There are 2380 top 10 lists a piece can occur in. Without further knowledge one would expect a piece to be in about 10 of the top 10 lists.

However, for G1 there are 275 songs which do not occur in any of the top 10 lists. For G1C this number is reduced to 174. For G1 there is one song which occurs in 253 of the top 10 most similar lists to all other pieces. This song is "Smudo schweift aus" by "Die Fantastischen Vier" (German hip hop group) from the album "Die 4. Dimension". The song has a duration of 3'47" of which the last 50 seconds are an instrumental. However, these 50 seconds are not considered for the similarity computation (only 2 minutes from the center are used from each piece). From a perceptual point of view there are no particularities which could explain the anomaly. The same song is only in 33 of the top 10 lists for G1C.

The most frequent song in the G1C top 10 lists appears 81 times. The song appears 125 times in the G1 top 10 lists (which is rank 3). The song is from the same album and has the title "Zu Geil für diese Welt". In total there are 5 songs by this group in the top 20 for G1.

Figure 2.22 shows for each piece in DB-L how many times it occurred in the top 10 lists for G1 and for G1C. As can be seen, the outliers are far less extreme for G1C. The reason for this is that the "always similar" problem is caused by the spectral similarity and is "fixed" to some extent by the features (FP etc.) which define a vector space. As such "always similar" pieces are likely to have a negative impact on any application based on audio similarity, any approach which reduces the effects is favorable.

2.2.7.2 Triangular Inequality

The triangular inequality is one of the properties of a metric space: $d(A, C) \leq d(A, B) + d(A, C)$. It does not hold for the spectral similarity measures. This restricts the techniques that can be used to efficiently index large music collections.

The extent to which the triangular inequality does not hold can be measured by computing the percentage of all triangles in a collection (DB-L in this case) that do not fulfill the inequality. Instead of computing all possible triangles 150000 triangles were randomly sampled. For G1 the inequality holds in average for 29% of the cases, and 36% for G1C. Again, this increase is favorable because it makes the similarity space easier to deal with (e.g. when developing heuristics for playlist generation).

However, the triangular inequality is not necessarily a concept humans apply for similarity judgments. For example, let A be a piece of classical music with a popular melody such as "Für Elise" by Beethoven. Let C be some extreme techno piece. Let B be an extreme techno piece with the melody of Für Elise. Obviously, A and B have some similarities, and so do B and C, However, A and C could hardly be any more dissimilar.

2.2.7.3 Always Dissimilar

Some pieces are totally different to all others, according to G1. This can be seen in the distance matrix shown in Figure 2.23 as black vertical and horizontal lines. These pieces originate from three albums. One album is from the genre a cappella: Golden Gate Quartet, When the Saints Go Marching In. The two other albums are from the genre jazz guitar: Barney Kessel, Solo; and Herb Ellis & Joe Pass, Two For The Road. About 1.5% of the songs in DB-L are extremely dissimilar. The fact that only so few pieces are effected makes it difficult to measure the effect in terms of the nearest neighbor genre classification evaluation procedure used later on.

The extremely high dissimilarity means that these pieces would seldomly or even never occur in automatically created playlists, in contrast to the "always similar" pieces. Although the combination of G1 with FP and other features reduces the effect, further work is needed to ensure that the distances are less extreme. One option might be to optimize the scaling of G1. To evaluate the effects it is necessary not only to consider the most similar piece to each piece but rather to evaluate if a similarity measure can retrieve all similar pieces in a collection.

2.3 Optimization and Evaluation

Different features have different advantages. Thus, it is straightforward to combine them. This leads to two questions. First, how can the performance of a (combined)

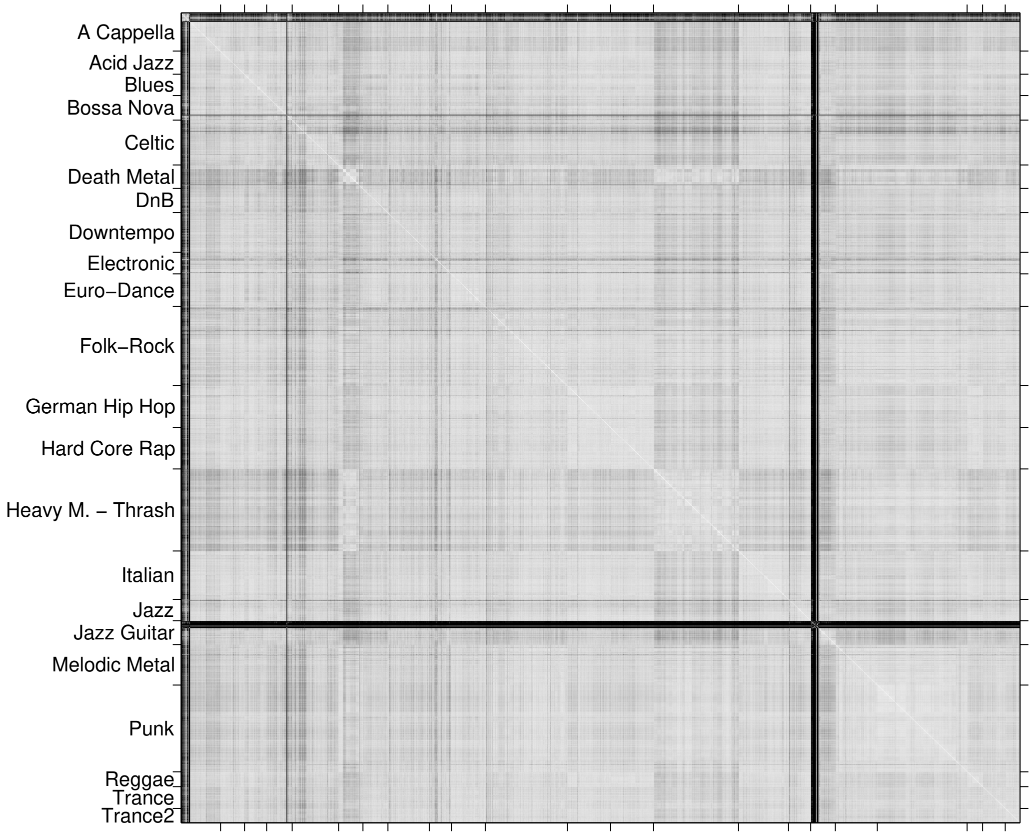


Figure 2.23: Distance matrix for DB-L computed with G1C.

similarity measure be measured? Second, how can the search space of all possible combinations be limited?

Measuring Similarity Measures?

In an ideal case, a listening test is used to assess the performance of a similarity measure. Such a test is presented in Subsection 2.3.6.3 to evaluate the best combination compared to a baseline. However, it is impracticable to use listening tests to evaluate thousands of different variations of similarity measures. A simple approach to evaluate combinations is to use a genre classification based approach. The assumption is that the most similar pieces to each piece belong to the same genre. The exact procedure and problems involved in this approach are described later on.

Limiting the Search Space?

The complete space of interesting combinations according to Equation 2.41 is huge. Given only 14 features and one spectral similarity measure means that there are 15 weights that can be set. (The weights are non negative and sum up to 1.) If the weights are set to multiples of 0.1 almost 2 million combinations are possible. For multiples of 0.05 the number of possible combinations is beyond 10^9. In this section the search space is limited by selecting the most interesting features. In particular, each feature is combined with the spectral similarity and only those which perform well are considered further.

Structure of this Section

The next subsection gives an overview of related work dealing with different approaches used to evaluate audio-based music similarity measures. Subsection 2.3.2 describes the procedure, in particular the genre-based evaluation approach. Subsection 2.3.3 presents the data used for the optimization and evaluations. In total there are 6 collections with a total of more than 20000 pieces and more than 60 genres. Subsection 2.3.4 describes the concept of an artist filter which should be used for the evaluation of audio-based genre classification and music similarity. Subsection 2.3.5 presents the optimizations of the parameters for the linear combination. Particular effort is made to avoid overfitting. Subsection 2.3.6 evaluates the best combination compared to a baseline (which is using only spectral similarity).

2.3.1 Related Work

Several approaches to compute audio-based music similarity have been published. However, evaluating these approaches is difficult. First, most implementations have not been made freely available. Second, in most cases it is not possible to share the music collections used for evaluation due to copyright restrictions. Third, results on different music collections (and genre taxonomies) are not comparable.

One solution has been to reimplement approaches by other authors. For example, in [PDW03b] five different approaches were implemented. However, there is no guarantee that these implementations are correct. In fact one approach ([LS01] and in particular the KL-divergence) was not implemented correctly in [PDW03b]. Another example is [BP03] where the authors reimplement one approach and developed their own variation.

An alternative is the collaboration of authors (e.g. [BLEW03]) or the use of creative commons music which is easily available (such as the Magnatune collection used for the ISMIR'04 contest), or collections especially created for such purposes such as the RWC music database [GHNO03]. For commercially interesting music (and thus not easily accessible music) one solution is a centralized evaluation system [DFT04] as used for the ISMIR'05 evaluation exchange (MIREX).

Logan & Salomon

Given different algorithms, or variations of the same algorithm, different procedures have been used for evaluation. One of the first was published by Logan & Salomon [LS01]. They presented three evaluations. First, they evaluated the optimum number of MFCCs. Second, they used a listening test to compare their best parameter settings against random shuffling. Third, they measured how well the original version of a song can be retrieved given a clipped version of the original as query.

They evaluated the optimum number of MFCCs as follows. For each song in the collection, they computed a list of 20 similar pieces. (The song to which the similarities are computed is referred to as seed). In these lists they counted the average number of songs (1) in the same genre as the seed, (2) from the same artist as the seed, and (3) on the same album as the seed. The correlation of the results for these 3 criteria is very high.

The listening test Logan & Salomon used was set up as follows. Two independent listeners where asked to rate 20 lists of similar songs (each with 20 songs). The 20 seed songs were randomly selected. For each song the listeners were requested to rate if they consider it similar to the seed or not (yes/no). The concept of similarity was only defined vaguely. Nevertheless, there was a high agreement. The listeners only disagreed in 12% of the cases. This consistency between listeners is confirmed by the results of the listening test presented in this thesis.

Logan & Salomon report the average number of similar songs (in the top 5, 10, 20 ranked positions) compared to random and show that their approach works significantly better. In particular, random finds in average 0.8 similar songs in the top 20 while their approach finds 8.2 songs.

Aucouturier & Pachet

In [AP02a] the authors hand-optimized the parameters such as the number of clusters and MFCCs. They report the following evaluation statistics. (1) They report the average number of songs in the same genre (for the closest 1, 5, 10, 20, 100 songs). (2) They measure the "overlap on the same genre". Given a seed song let d_m be the mean distance between songs from the seed's genre. Let n_1 be the number of pieces not belonging to pieces from seed's genre, and let n_2 be the number of pieces not belonging to seed's genre and having a distance to the seed which is smaller than d_m. The overlap is defined as the ratio: n_2/n_1. In addition they report the "overlap on a different genre" which is the proportion of songs in the seed's genre which have a distance to the seed which is larger than d_m. (3) They report precision and recall.

Furthermore, in addition to the statistics the authors briefly report a listening test with 10 listeners which were asked to rate songs by similarity. In particular, given a seed they were asked to rate the similarity of two songs. One of these was closer to the seed and the other was further away (according to their similarity measure). The listeners agreed in 80% with the ordering of the similarity measure.

In [AP04a] the authors report results of an extensive evaluation. In particular, they analyzed the impact of parameters such as the number of MFCCs, the number of Gaussians used for the GMM, and the window size for the MFCC computation. To measure the differences they use the R-precision. (That is, the precision within the R closest songs to the seed, where R is the number of relevant songs.)

In [AP04b] the authors describe the framework and tools used to obtain the results published in [AP04a]. The authors identify two types of similarity evaluations: (1) rankings and (2) classes. With rankings they refer to the style of evaluations conducted in, e.g., [BLEW03]. This requires similarity data which is usually difficult to gather. The second type is the approach they use themselves and has been used, e.g., in [LS01] and [PDW03b]. The authors suggest using the TREC[16] style evaluation metrics for evaluations of this type. This is done, e.g., in [SK05] and [LR05]. The use of precision and recall statistics is also discussed in [Bau05].

Others

In [PDW03b] an evaluation of 5 different similarity measures was presented using the following cluster quality criteria. First, the average distance between all pieces was computed. This average distance was then compared to the average distance within a group (groups were defined either in terms of artists, genres, albums, tones, or styles). A ratio of one means that the distance between arbitrary pieces and members of a group are about the same. On the other hand, if the distances within the groups are very small compared to the overall average distances then the groups are better distinguishable from each other.

[16] http://trec.nist.gov

One of the outcomes of this evaluation was that spectrum histograms performed very well. However, in [Pam04] where the same 5 measures were evaluated (using the R-precision) the results were very different. In particular, it seems that the cluster quality is a suboptimal criteron to evaluate similarity measures.

In [AHH+03] the authors briefly report a listening test where they compare their algorithm against random shuffling. They selected 20 random seeds and computed the 5 nearest neighbors to each seed. In addition, for each seed they randomly selected one piece from the collection, and the piece furthest away from the seed (according to the similarity measure). They asked 10 subjects to rate the similarity of these 7 pieces. As a result the authors are able to show that their approach works better than random.

The measurable difference to a purely random approach depends on the music collection used. For example, [PE02] evaluated their playlist generation approach against random shuffling on a collection consisting of only jazz music. The differences they observed are much smaller than those reported in [AHH+03].

Ultimately, evaluating similarity measures within their application context is the only way of finding out if they are useful or not and if differences are significant or not. Application based evaluations have been reported, e.g., in [BH04; VP05; GG05; PPW05a; PPW05c].

In [LEB03; BLEW03] the authors suggest a solution to the problem of not being able to share music collections for evaluation purposes. In particular, they recommend sharing features extracted from the audio instead of the audio data itself. Furthermore, they use subjective similarity data gathered from surveys, experts, playlists, and directly from the contents of user collections to evaluate similarity measures. One major disadvantage of such data is its sparsity. That is, similarity connections are usually only defined for a tiny fraction of all possible connections.

In [PFW05b] the criterion to evaluate the measures is the accuracy of a nearest neighbor classifier in combination with leave-one-out cross-validation. This means that for each song in the collection the nearest neighbor is computed. The average number of times that this nearest neighbor is from the same genre as the seed is the accuracy of the "classifier". In addition an artist filter is used which will be discussed later on. One of the advantages of using classification accuracies to evaluate similarity measures is that it allows comparison to the large number of publications on genre classification.

2.3.1.1 MIREX

Recently, within the MIR community there has been a strong effort to standardize evaluations. As part of these efforts, there have been contests to evaluate the best genre classification and artist identification algorithms at ISMIR'04 and at ISMIR'05 where the contest was named MIREX. MIREX stands for the Music

Information Retrieval EXchange.[17] Of particular interest is that at MIREX'06 there will be a music similarity task for the first time. The discussion on the optimum evaluation procedure is ongoing at the time of writing this thesis.

Before describing the 2004 and 2005 contests the following paragraphs discuss the relationship between genre classification and artist identification, and how genre classification can be used to evaluate similarity measures.

Genre Classification and Artist Identification

In 2004 and 2005 one of the tasks has been audio-based artist identification. The task is to identify the artist given a piece. The advantage of artist identification is that there is an objective ground truth. However, it is not so clear what real world applications could be. Although rich metadata is rare, tracks usually have an artist name assigned to them. If a track does not have any metadata associated to it then an obvious approach would be to use fingerprinting technology to identify the track and obtain its metadata.

Compared to genre classification there is an obvious technical difference. The artist identification task has more and smaller categories. This can be a problem for some classifiers whose complexity depends on the number of categories.

In general, an algorithm that performs well on artist identification might not perform well on genre classification. In particular, this can be the case if the algorithm focuses on production effects or a specific instrument (or voice) which distinguishes an artist (or even a specific album). That is, if the algorithm focuses on characteristics which a human listener would not consider relevant for defining a genre.

However, genre classification is often evaluated on music collections where all pieces from an artist have the same genre label without using an artist filter. An artist filter ensures that given a piece to classify, no other pieces from the same artist have been used for training the classifier. An algorithm that can identify an artist would also perform well on genre classification if no artist filter is used. In particular, in such a case the genre classification and artist identification tasks are far more similar than they might appear at first. In 2004 and 2005 no artist filters were used and all pieces from an artist were assigned to the same genre.

Similarity and Genre Classification

One of the simplest ways to evaluate similarity measures is through genre classification. The assumption is that very similar pieces belong to the same genre. A classifier used to evaluate similarity measures should be directly based on the similarity computations. A straightforward choice is to use a nearest neighbor classifier.

[17] http://www.music-ir.org/mirexwiki

The question which remains is if a classifier should be allowed to fine tune some similarity parameters based on the training data or not. One approach to answer this question is to look at specific application scenarios. For example, if the similarity measure is applied to a collection which has already been classified (manually) into genres, then such optimizations make sense. In this chapter this assumption is not made.

ISMIR 2004

The first audio-based genre classification contest was organized by the Music Technology Group (MTG) at Universitat Pompeu Fabra (Barcelona, Spain) in 2004. MTG supplied a training set on which participants could train their classifiers. The participants would then submit a classifier which would run on the test set located at MTG. The data used was the Magnatune[18] collection, which is available under the creative commons license.

There were a total of 5 groups which participated in the contest. An approach based on G30 won the contest with 84% classification accuracy, an approach based on Fluctuation Patterns scored 70%. The results are available online.[19]

ISMIR 2005 (MIREX)

Based on the experiences gathered in 2004 a Java-based M2K (Music to Knowledge)[20] framework was designed based on D2K (Data to Knowledge)[21] to standardize the interface for submissions within the IMIRSEL project [DFT04].

For the genre classification and artist identification contests two collections were used. Namely, the Magnatune as in 2004, and in addition the USPOP 2002 collection (which has been previously used e.g. in [WL02]). As the USPOP collection contains copyright protected music, it was not possible to share it among the participants. Instead, the participants needed to submit algorithms that would be trained on the computers at IMIRSEL. The Magnatune collection was used by some groups to optimize their submissions, and some also had access to USPOP.

As in 2004 the main criteria was the classification accuracy. For the Magnatune collection a hierarchical taxonomy was used to differentiate between different types of errors (e.g. confusing jazz and blues versus jazz and heavy metal). Of particular interest are also the reported computation times. The measured differences were of a factor 20 and more. Participants were given 24 hours for their algorithms to complete. Memory usage was not reported.

A total of 12 groups participated in the genre classification and 7 in the artist identification task which is a large increase compared to 2004. Participants were

[18] http://www.magnatune.com
[19] http://ismir2004.ismir.net/genre_contest/results.htm
[20] http://www.music-ir.org/evaluation/m2k
[21] http://alg.ncsa.uiuc.edu/do/tools/d2k

	Participant	Hierarch.	Norm. Hierarch.	Raw	Norm. Raw	Time [hh:mm]	CPU Type
1	Bergstra et al. (2)	77.75	73.04	75.10	69.49	–	–
2	Bergstra et al. (1)	77.25	72.13	74.71	68.73	06:30	B
3	Mandel & Ellis	71.96	69.63	67.65	63.99	02:25	A
9	Pampalk	69.90	70.91	66.47	66.26	00:55	B

Table 2.2: Partial genre classification results for the Magnatune collection. 1005 tracks were used for training, 510 tracks for testing, and about seven genres needed to be classified. CPU Type A is a system with WinXP, Intel P4 3.0GHz, and 3GB RAM. CPU Type B is a system with CentOS, Dual AMD Opteron 64 1.6GHz, and 4GB RAM.

allowed to submit more than one algorithm. The details of the results are online.[22]

Some of the results are given in Tables 2.2–2.5. Normalized accuracies are computed by weighting the accuracy with respect to each genre equally (and thus ignoring the uneven distribution of genres). The difference between hierarchical and raw is that with the former the hierarchical taxonomy was used to weight errors differently.

The winning algorithm for the genre classification task was submitted by Bergstra et al. [BCE05] and is based on a powerful Ada Boost classifier. However, it is not straightforward to adapt the approach for similarity computations. The winning algorithm for artist identification submitted by Mandel & Ellis [ME05] is based on the G1 spectral similarity described earlier. However, instead of using a nearest neighbor classifier to evaluate the performance in terms of a similarity measure, a SVM variant is used.

The algorithm submitted by the author of this thesis was based on G30S combined with FP, FP Gravity, and FP Focus (this combination is referred to as M'05 for details see [Pam05]). It was the only algorithm using a nearest neighbor classifier for the genre classification task and one of two for artist identification task. As will be shown later in this section (Table 2.11), M'05 performs about the same as G1 on the Magnatune collection when both use a nearest neighbor classifier.

2.3.2 Procedure

In this section two approaches are used. One is based on genre classification. It serves to explore large parameter spaces. The other is based on a listening test and is used to evaluate if the improvements are significant to human listeners.

[22] http://www.music-ir.org/evaluation/mirex-results

	Participant	Raw	Norm. Raw	Time [hh:mm]	CPU Type
1	Bergstra et al. (2)	86.92	82.91	–	–
2	Bergstra et al. (1)	86.29	82.50	06:30	B
3	Mandel & Ellis	85.65	76.91	02:11	A
4	Pampalk	80.38	78.74	00:52	B

Table 2.3: Partial genre classification results for the USPOP'02 collection. 940 tracks were used for training, 474 tracks for testing, and about four genres needed to be classified.

	Participant	Raw	Norm. Raw	Time [hh:mm]	CPU Type
1	Bergstra et al. (1)	77.26	79.64	24:00	B
2	Mandel & Ellis	76.60	76.62	03:05	A
3	Bergstra et al. (2)	74.45	74.51	–	–
4	Pampalk	66.36	66.48	01:11	B

Table 2.4: Partial artist identification results for the Magnatune collection. 1158 tracks were used for training and 642 for testing.

	Participant	Raw	Norm. Raw	Time [hh:mm]	CPU Type
1	Mandel & Ellis	68.30	67.96	02:51	A
2	Bergstra et al. (1)	59.88	60.90	24:00	B
3	Bergstra et al. (2)	58.96	58.96	–	–
4	Pampalk	56.20	56.03	01:12	B

Table 2.5: Partial artist identification results for the USPOP'02 collection. 1158 tracks were used for training and 653 for testing.

Nearest Neighbor Genre Classifier

To evaluate various parameter settings a genre-based approach is used. In particular, the classification accuracy of a nearest neighbor classifier is evaluated using leave-one-out cross-validation (as described in the related work). The accuracies are not normalized with respect to class probabilities. The basic assumption is that very similar pieces belong to the same genre.

The main advantage is its simplicity. Genre tags for music are readily available, the results are easy to interpret, and are comparable to related work published on genre classification. (Which is particularly interesting as the amount of work published on genre classification is much larger than work directly related to similarity measures.) However, evaluating only the nearest neighbor also has limitations. No distinction is made if all pieces which are close to the seed are perceptually similar, and no distinction is made if all pieces perceptually similar to the seed are also close to the seed.

There are two issues to consider when arguing against using the nearest neighbors only. First, not all songs in a genre are similar to each other. Thus, it is not possible to assume that all pieces from the same genre are similar to the seed. If similar songs were defined manually an evaluation which considers more than just the nearest neighbor would make a lot more sense.

Second, assuming that the application is playlist generation, one might only need very few similar songs to each song to create an interesting "chain" of songs. That is, depending on the application, it might not be necessary to find all songs similar to a single seed.

Nevertheless, the final results in this section indicate that considering more than only the nearest neighbor leads to more accurate evaluations. However, the question remains if differences which can only be measured in such a way are significant.

Genre Taxonomies

Much can be said about the usefulness or uselessness of music genres. The fact is that genre taxonomies are inconsistent and have a number of other limitations (which are discussed, e.g., in [PC00]). An obvious issue is that many artists have a very individual mix of several styles which is often difficult to pigeonhole.

However, genres are widely used to manage large music collections, and genre labels for artists are readily available. Furthermore, as will be discussed in Subsection 2.3.6 the quality of most genre annotations available is significantly better than the quality which current genre classifiers and similarity measures can achieve. Thus, there is still a lot to be learned from genre annotations.

Avoiding Overfitting

Avoiding overfitting is one of the main topics throughout this section. Overfitting occurs when the parameters of a model are optimized in such a way that they produce good results on the specific data used to optimize them, but poor results in general. The following strategies are applied to avoid overfitting:

A. Multiple Collections
To avoid overfitting 6 different music collections which are organized according to different taxonomies are used. Two collections are used to optimize the parameters and four collections to evaluate the results and test how they generalize. The approach chosen in this thesis is very conservative. Alternatively, one could argue that overfitting is indeed context specific learning. However, such assumption would require an appropriate evaluation.

B. Artist Filter
This important topic is discussed in detail in Subsection 2.3.4. The basic idea is to reduce the opportunities an algorithms has to optimize the parameters to information that is perceptually not relevant.

C. Cross Checking Spectral Similarity
In Subsection 2.2.3 three different approaches to compute spectral similarity were described. These are more or less identical in terms of the similarity aspect they describe. One of these approaches might be better than the others. However, in general any non-systematic deviation points towards an insignificant variance. In particular, any optimizations of G1 are cross checked with G30S and vice versa to see whether the optimization can be generalized.

Optimization and Evaluation Outline

The procedure chosen in this thesis is as follows. First, features are selected to reduce the hypothesis space. Second, the parameters of the combination are optimized for the selected features using two music collections. Third, the parameters are tested on different collections. Finally, a listening test is used to test whether the differences measured through the genre classification approach are significant.

2.3.3 Data

For the optimization and evaluation a total of six collections are used. Each has its own genre taxonomy. There are more than 20000 pieces in total which are assigned to more than 60 different genres. The term genre is used very flexibly, for example, “genres” include: “others” or “romantic dinner”. Statistics of the collections are given in Table 2.6. A list of genres with the number of pieces and artists associated to each is given in Table 2.7.

	Genres	Artists	Tracks	Artists/Genre Min	Artists/Genre Max	Tracks/Genre Min	Tracks/Genre Max
DB-S	16	63	100	2	7	4	8
DB-L	22	101	2381	3	6	42	255
DB-MS	6	128	729	5	40	26	320
DB-ML	10	147	3249	2	40	22	1278
DB-30	29	528	1470	4	48	19	80
DB-XL	17	566	15335	5	113	68	3330

Table 2.6: Statistics of the six collections.

The collections DB-S, DB-L DB-MS, and DB-ML have been previously used in [PFW05b]. Deviations in the exact number of pieces are caused by the MP3 decoder used and whether all files could be decoded correctly. For the experiments reported in this thesis the robust MAD[23] decoder was used. Furthermore, in some cases very short (e.g. shorter than 15 seconds) pieces were removed from the collections.

In-House Small (DB-S)

The smallest collection consists of 100 pieces. The collection also includes one non-music category, namely speech (German cabaret). This collection has a very good (i.e low) ratio of tracks per artist. However, due to its size the results need to be treated with caution. In this section DB-S is mainly used to demonstrate overfitting effects.

In-House Large (DB-L)

This collection is hierarchically structured according to genre/artist/album. All pieces from an artist (and album) are assigned to the same genre, which is questionable but common practice. Two pieces overlap between DB-L and DB-S, namely Take Five and Blue Rondo by the Dave Brubeck Quartet. The genres are user defined and inconsistent. In particular, there are two different definitions of trance. Furthermore, there are overlaps, for example, jazz and jazz guitar, heavy metal and death metal etc. DB-L is on of the two collections used for optimization. In addition, the normalization parameters (for the variance normalization which is necessary to combine different similarity measures) are computed based on DB-L.

Magnatune Small (DB-MS)

This collection was used as training set for the ISMIR'04 genre classification contest. The music originates from Magnatune[24] and is licensed as creative commons.

[23]http://www.underbit.com/products/mad
[24]http://www.magnatune.com

MTG[25] compiled the collection. Although it is a subset of DB-ML it is particularly interesting as it has been made available to the research community. Furthermore, to some extent it can be used to compare results to those of the ISMIR'04 contest. However, while the code which won the contest achieves 79% accuracy on this training set, the accuracy on the test set was 84%. This is probably related to the artist filter issue discussed below, as half of the pieces of each album were split between training and test set and all pieces from an artist belong to the same genre. The genre labels are given on the Magnatune website. The collection is very unbalanced. Most pieces belong to the genre classical and a large number of pieces in world sound like classical music. Some of the original Magnatune classes were merged by MTG due to ambiguities and the small number of tracks in some of the genres. DB-MS is one of the two collections used to optimize the parameters.

Magnatune Large (DB-ML)

DB-ML is a superset of DB-MS. The number of artists is not much larger than in DB-MS and the genres are equally unbalanced. The genres which were merged for the ISMIR'04 contest are separated. DB-ML is used to see whether the parameters optimized using DB-MS perform as well on a larger collection with very similar characteristics.

In-House 30 Seconds (DB-30)

All pieces in this collection have a length of 30 seconds. This length is of interest as many online music shops offer their customer 30 second previews. The genre tags for this collection are those shown to the customer of an online music store. Some of these genres are obviously very difficult if not impossible to distinguish for a simple audio-based similarity measure. For example, the genre named Christmas contains Christmas songs which are primarily recognizable by their lyrics. DB-30 is of particular interest as it reflects the taxonomies currently in use by music distributers and points out the limits.

In-House Extra Large (DB-XL)

This is the largest collection used. The genre labels are assigned on a piece level according to Gracenote.[26] DB-XL is used to test whether the optimizations can be generalized or not. Due to its size it is by far the most interesting collection.

2.3.4 Artist Filter

A key issue in evaluating music genre classification is the use of an artist filter [PFW05b]. An artist filter ensures that the training and test set contain

[25] http://www.iua.upf.es/mtg
[26] http://www.gracenote.com

DB-S	alternative (6/4), blues (8/2), classic orchestra (6/6), classic piano (7/2), dance (6/4), eurodance (5/5), happy sound (6/3), hard pop (5/2), hip hop (7/3), mystera (8/2), pop (6/5), punk rock (8/7), rock (4/4), rock & roll (6/5), romantic dinner (6/6), speech (6/3)
DB-L	a cappella (112/4), acid jazz (68/4), blues (63/4), bossa nova (72/4), celtic (132/5), death metal (69/4), DnB (71/5), downtempo (117/4), electronic (63/4), euro-dance (96/6), folk-rock (233/5), German hip hop (123/6), hard core rap (122/5), heavy metal/thrash (241/5), Italian (142/5), jazz (63/4), jazz guitar (70/5), melodic metal (119/4), punk (255/6), reggae (44/3), trance (64/5), trance2 (42/4)
DB-MS	classical (320/40), electronic (115/30), jazz/blues (26/5), metal/punk (45/8), pop/rock (101/26), world (122/19)
DB-ML	ambient (143/6), classical (1278/40), electronic (459/30), jazz (104/5), metal (116/6), new age (191/13), pop (22/2), punk (64/2), rock (383/24), world (489/19)
DB-30	ambient/new age (65/12), blues (42/11), children/others (52/12), Christmas (39/22), classical (44/20), country (59/18), dance (56/28), Dutch (19/4), easy listening/oldies (69/33), electronic (66/17), folk (52/16), German pop (60/23), gospel/Christian music (59/16), indie/alternative (28/15), instrumental (60/7), jazz (49/16), latin (52/9), Italian (22/8), pop (80/48), others (58/13), rap/hip hop (23/12), reggae (30/7), rock (50/36), Schlager (69/21), soul/R&B/funk (44/28), soundtracks (58/29), speech (75/10), Variété Française (33/24), world (57/31)
DB-XL	alternative & punk (2076/95), blues (254/13), Christmas (68/5), classical (626/25), country (1968/88), easy listening (119/8), electronica/dance (1861/93), folk (158/11), hip hop/rap (682/28), jazz (3331/113), latin (366/18), new age (263/16), pop (551/32), R&B (1097/50), reggae (131/6), rock (1491/53), world (294/17)

Table 2.7: List of genres for each collection. For each genre the first value in the brackets is the number of pieces, the second value is the number of artists.

		DB-S	DB-MS				DB-ML				DB-L			
	AF	1	1	5	10	20	1	5	10	20	1	5	10	20
G30	−	52	79	77	74	70	80	79	78	75	71	68	67	63
	+	29	64	67	67		56	59	60		27	28	30	31
G30S	−	53	78	75	71	68	79	77	76	73	71	69	68	65
	+	29	62	65	63		51	55	57		25	28	30	31
G1	−	56	78	76	74	70	81	79	77	74	76	73	72	69
	+	31	63	65	66		54	57	59		28	30	31	31
FP	−	47	61	62	64	64	64	64	64	63	45	45	44	43
	+	30	54	56	60		50	53	54		24	23	23	22

Table 2.8: Leave-one-out cross-validation results using a k-NN classifier with $k = 1, 5, 10, 20$. If in any of the genres less than k items remained after applying the artist filter, then the k value was not evaluated and left blank.

different artists. In particular, when using spectral information the classification accuracies are over estimated otherwise. Furthermore, an algorithm optimized without artist filter might focus on using perceptually not so relevant information such as production effects. Thus, an artist filter is necessary to measure generalization capabilities more accurately.

In addition, one can argue that filtering results from the same artist is important in most application scenarios for similarity measures. This is particularly true for recommendation, where one can expect the user to know the artists from the piece which the user selected as query. Furthermore, it is also true for automatically created playlists, where in general a good playlist is not one which contains only pieces from the same artist.

Table 2.8 shows the results of a nearest neighbor classification (measured using leave-one-out cross-validation) for four of the similarity measures described earlier. AF+ marks lines where an artist filter was used AF− marks lines evaluated without artist filter. As can be seen, the difference between AF+ and AF− almost reaches 50 percentage points for G1 and DB-L. This large difference is partly due to the low number of artists per genre. However, even for DB-XL (results in Table 2.11) the difference between AF+ and AF− is still 25 percentage points.

In addition to the nearest neighbor also k-NN accuracies are given for k equals 5, 10, and 20 to study the effect of using (or not using) an artist filter more accurately. Particularly interesting is that the AF+ accuracies consistently decrease with increasing k while the AF− accuracies increase. This confirms that results measured without artist filter are not reliable performance indicators. Noticeable is also that FP is not as strongly effected. This suggests that artist-based overfitting is particularly a problem when using spectral similarity.

Conclusions

One obvious conclusion is that all results reported in the remainder of this thesis (unless stated otherwise) use an artist filter. Another conclusion from Table 2.8 is that no further G30 results are computed. The results show a high correlation with G1 and G30S (Figures 2.20 and 2.21), the performance is similar (Table 2.8), and G30 is more than a factor 100 slower than G30S and more than a factor 1000 slower than G1 (Table 2.1).

2.3.5 Optimization

The goal of this subsection is to optimize the weights of the linear combination. First, the size of the parameter space is reduced by selecting the most promising features for the combination. Second, the results of an exhaustive search in the remaining space are presented.

2.3.5.1 Combining Two (Feature Selection)

Section 2.2 describes a total of 14 features which can be combined (according to Equation 2.41) with a spectral similarity measure. Assuming that all weights are in the range of 0 to 1 with a step size of 0.1 means that nearly 2 million different combinations need to be explored, which is too computationally expensive. Thus, it is necessary to select a subset of features for the combination.

To select a subset the 14 features are combined individually with spectral similarity. Those which perform best are used for further analysis. To avoid overfitting effects the combinations are measured with G1 and G30S and the results are averaged. In addition, two collections are used (DB-MS and DB-L).

Figure 2.24 shows all the combination results. In addition to combining the 14 features with spectral similarity, also G1 is combined with G30S and vice versa (this combination is abbreviated with "Spec. Sim."). Based on the assumption that combining these does not introduce new information, this serves as a baseline for significant improvements.

The last column in Figure 2.24 shows the classification accuracies if the features are used by themselves. FPs achieve the highest individual accuracy with 54% on DB-MS and 24% on DB-L. Noticeable are that most combinations can improve the spectral similarity only marginally. In several cases there is no improvement. Using the combination of the G1 and G30S as significance baseline, improvements of up to 1 percentage point can be considered insignificant on DB-MS.

Furthermore, noticeable are the smooth changes of the classification accuracy between similar weights. That is, changing the weights by 10 percentage points does generally not have a large impact on the accuracies. The exception is when the weight of spectral similarity is dropped from 10% to 0%. At this point a large decrease in accuracy is noticeable for most of the features.

G30S/DB−MS

	100	90	80	70	60	50	40	30	20	10	0
RMS	62	62	61	60	60	59	59	58	56	57	44
ZCR	62	62	62	60	58	58	57	55	54	53	29
Noisiness	62	62	62	61	61	59	57	56	55	50	29
SC	62	62	62	61	60	59	59	59	56	54	40
Avg. Loudness	62	62	63	61	60	58	57	55	54	53	44
Perc.	62	65	66	66	64	63	60	60	58	57	45
FP	62	61	64	64	63	63	61	62	62	61	54
FP Gravity	62	66	66	66	65	64	63	61	61	60	47
FP Focus	62	61	62	61	60	58	55	54	51	46	24
FP Max	62	62	62	60	59	58	59	59	57	55	34
FP Sum	62	61	61	61	61	61	60	59	57	52	27
FP Bass	62	63	63	64	63	63	62	61	61	59	34
FP Aggr.	62	62	62	62	61	61	62	60	59	54	42
FP DLF	62	63	64	64	63	62	61	60	58	56	31
Spec. Sim.	62	63	63	63	63	64	63	63	64	64	63

G1/DB−MS

	100	90	80	70	60	50	40	30	20	10	0
RMS	63	63	62	59	60	60	59	58	57	55	44
ZCR	63	60	60	60	59	58	56	55	54	52	29
Noisiness	63	62	62	61	60	59	58	56	54	51	29
SC	63	62	61	60	60	60	59	58	55	53	40
Avg. Loudness	63	61	59	57	56	56	57	56	55	52	44
Perc.	63	63	63	62	61	60	60	59	58	58	45
FP	63	63	63	64	64	65	65	64	63	60	54
FP Gravity	63	67	66	65	65	64	64	64	64	60	47
FP Focus	63	65	63	61	59	56	54	52	49	46	24
FP Max	63	63	61	62	62	60	58	58	56	54	34
FP Sum	63	63	61	60	60	58	58	57	55	52	27
FP Bass	63	62	64	63	63	63	61	62	61	57	34
FP Aggr.	63	63	62	60	60	60	61	59	57	53	42
FP DLF	63	65	64	64	63	62	63	62	60	56	31
Spec. Sim.	63	64	64	63	63	64	63	63	63	63	62

G30S/DB−L

	100	90	80	70	60	50	40	30	20	10	0
RMS	25	25	23	23	22	22	21	20	19	17	5
ZCR	25	26	26	26	26	25	25	24	23	22	8
Noisiness	25	29	28	27	27	27	26	25	23	21	8
SC	25	28	29	28	27	27	26	26	25	23	10
Avg. Loudness	25	27	27	26	25	25	25	23	22	20	9
Perc.	25	29	28	28	27	27	26	24	23	20	8
FP	25	30	33	32	33	31	31	30	29	27	24
FP Gravity	25	31	32	31	30	30	29	28	26	24	8
FP Focus	25	26	25	25	24	24	24	23	21	19	5
FP Max	25	26	25	25	24	24	23	23	22	20	8
FP Sum	25	27	28	27	26	26	25	25	23	21	11
FP Bass	25	28	27	27	27	27	27	26	24	23	9
FP Aggr.	25	27	27	27	26	25	25	24	22	20	7
FP DLF	25	27	27	26	26	26	25	24	22	21	5
Spec. Sim.	25	25	25	26	26	26	27	27	27	27	28

G1/DB−L

	100	90	80	70	60	50	40	30	20	10	0
RMS	28	24	22	21	20	19	18	18	17	15	5
ZCR	28	26	25	24	24	24	24	22	19	17	8
Noisiness	28	29	29	28	26	26	26	25	23	21	8
SC	28	28	28	28	27	27	26	26	24	21	10
Avg. Loudness	28	28	26	25	25	24	23	22	20	18	9
Perc.	28	29	27	26	25	24	22	22	21	18	8
FP	28	31	32	32	32	32	30	29	28	26	24
FP Gravity	28	30	29	28	28	27	27	25	24	22	8
FP Focus	28	27	26	24	24	23	22	21	18	16	5
FP Max	28	28	27	26	25	25	24	23	21	20	8
FP Sum	28	28	28	27	26	25	25	23	23	21	11
FP Bass	28	27	27	27	27	27	26	25	23	20	9
FP Aggr.	28	27	26	26	25	24	23	22	21	18	7
FP DLF	28	27	27	26	25	24	23	22	20	19	5
Spec. Sim.	28	27	27	27	27	26	26	26	25	25	25

Figure 2.24: The results of combining two measures for DB-MS and DB-L. All numbers are given in percent. The numbers below the table show the mixing coefficients. The first column uses only (100%) spectral similarity, the last column uses none (0%). For example, in the table in the upper left (DB-MS/G30S) the fourth column in the second row is the accuracy for combining 30% ZCR with 70% of G30S.

		DB-MS			DB-L			G30S	G1
	Overall $\sum$	G30S	G1	$\sum$	G30S	G1	$\sum$	$\sum$	$\sum$
FP Gravity	17.5	3.8	4.5	8.4	6.6	2.5	9.1	10.4	7.0
FP	16.1	1.6	2.3	4.0	7.9	4.2	12.1	9.5	6.5
Perc.	8.4	3.3		3.3	4.1	1.0	5.1	7.4	1.0
Noisiness	5.9				4.4	1.6	5.9	4.4	1.6
FP Bass	5.5	1.6	1.1	2.7	2.7		2.7	4.4	1.1
FP DLF	5.4	1.9	1.8	3.7	1.7		1.7	3.6	1.8
Spec. Sim.	5.2	1.5	1.1	2.6	2.6		2.6	4.1	1.1
SC	3.7				3.5	0.2	3.7	3.5	0.2
Avg. Loud.	3.3	0.5		0.5	2.0	0.8	2.8	2.6	0.8
FP Sum	3.2				3.0	0.2	3.2	3.0	0.2
FP Focus	2.7		1.8	1.8	0.9		0.9	0.9	1.8
FP Aggr.	2.1				2.1		2.1	2.1	
FP Max	1.7				1.0	0.7	1.7	1.0	0.7
ZCR	1.3				1.3		1.3	1.3	
RMS	0.1				0.1		0.1	0.1	

Table 2.9: Maximum increase in accuracy (absolute percentage points) per feature in combination with spectral similarity. Zero values are omitted.

Based on these results the 14 features are ranked in Table 2.9. For each feature the maximum possible improvement is considered. Particularly interesting is that the ranking does not correspond to the individual performance of the features. For example, the individual performance of RMS is higher than the individual performance of several of the other features. However, RMS is ranked last in Table 2.9. Every improvement below the improvement of combining G1 with G30S can be considered insignificant. Particularly noticeable are the poor performances of RMS, ZCR, and FP Focus.

FP and FP Gravity are ranked top if either the average performance per collection is considered, or the average performance per spectral similarity measure. Except for those features which completely fail it is not so obvious which of the other features are useful. Some work well combined with G1 (e.g. FP DLF) others with G30S (e.g. Percussiveness). Some show significant improvements only on DB-MS (e.g. FP Focus) others on DB-L (e.g. Noisiness). For simplicity, the top 6 features (all ranked above the G1-G30S combination) are selected for further optimizations. These features are FP Gravity, FP, Percussiveness, Noisiness, and FP Bass.

2.3.5.2 Combining Seven

There are 8008 possible combinations to consider given the 6 features selected above, plus one spectral similarity measure, and using weights with a step size of

0.1. The outcome of the evaluation of these 8008 combinations is summarized in Figure 2.25.

The best combination weight for spectral similarity (for all 4 variations) is around 60-80%. The largest increase occurs for DB-L with G30S where the increase is more than 10 percentage points (compared to using only G30S). Particularly, interesting is that for DB-L there are combinations without spectral similarity which achieve the same or better performance as the spectral similarity by itself. For DB-MS there are combinations which use only 10% spectral similarity and outperform spectral similariy. Finding combinations which do not use spectral similarity can be of interest if a vector space is a requirement of the application (or if computation time is very critical).

Furthermore, noticeable are the smooth changes in accuracy on the x-axis except between 10% and 0% spectral similarity. Features which do not have the maximum value in the first column require a minimum contribution greater than 0 to achieve the best results. For example, for DB-MS with G30S at least 10% contribution of Gravity and DLF are required to achieve 68% accuracy.

Table 2.10 lists the 10 combinations with the highest score. The score of a combination is computed as mean absolute percentage increase compared to the baseline (i.e., using only spectral similarity). The highest score is an average increase of 6.14 percentage points. Considering how much room there is for improvements this is rather disappointing.

Noticeable are that the best combinations frequently include FP, FP Gravity, and FP Bass. However, in most cases their contribution is not higher than 10%. Very interesting is that the differences between the highest possible scores and the highest average score is not very large. The largest difference is 1.2 percentage points for G1 and DB-L. These small differences do not justify using specifically optimized weights for each collection. The best combination with 10% FP + 10% FP Gravity + 10% FP Bass + 70% G1 will from now on be referred to as G1C (where the C stands for combined).

A question which remains open is how significant these improvements are. Figures 2.26–2.28 give an impression of the variance one should expect. Figure 2.26 shows that combinations with 0 contribution of spectral similarity score consistently worst. This is clearly visible by the zero variance for the combinations from about rank 5000–8008. (The reason for the zero variance is that there is no difference between G1 or G30S combinations if the contribution of the spectral similarity is 0.)

The deviations between G1 and G30S are relatively small compared to the performance increase. At least the top 500 combinations seem significantly better than the baseline. These combinations have in common that they use at least 20% spectral similarity.

Figure 2.27 shows even lower variances for DB-L (compared to DB-MS). Noticeable is that several 0% spectral similarity combinations score better than the 100% baseline. At least the top 1000 combinations seem significantly better than

G30S/DB–MS

	100	90	80	70	60	50	40	30	20	10	0
Spec. Sim.	62	66	67	68	68	68	68	67	66	65	58
Noisiness	68	67	66	65	65	64	61	59	57	50	29
Perc.	68	67	67	66	66	63	62	62	60	57	45
FP	68	68	67	67	67	66	65	63	63	61	54
FP Gravity	67	67	68	68	68	66	64	63	61	60	47
FP Bass	68	68	67	67	67	66	65	64	63	59	34
FP DLF	67	68	68	68	67	66	66	63	60	56	31
	0	10	20	30	40	50	60	70	80	90	100

G1/DB–MS

	100	90	80	70	60	50	40	30	20	10	0
Spec. Sim.	63	67	68	68	67	67	67	67	65	64	58
Noisiness	68	66	65	65	64	63	60	59	57	51	29
Perc.	68	67	67	66	66	65	64	62	60	58	45
FP	68	68	67	66	66	65	65	64	63	60	54
FP Gravity	67	68	67	67	66	65	64	64	64	60	47
FP Bass	68	68	67	67	66	65	65	64	63	57	34
FP DLF	68	68	67	67	67	65	65	63	60	56	31
	0	10	20	30	40	50	60	70	80	90	100

G30S/DB–L

	100	90	80	70	60	50	40	30	20	10	0
Spec. Sim.	25	31	34	35	36	35	35	34	34	32	28
Noisiness	35	36	34	34	33	32	30	29	28	21	8
Perc.	36	35	34	33	32	31	30	29	27	20	8
FP	35	36	36	35	34	33	32	32	30	27	24
FP Gravity	35	36	35	35	33	33	31	29	26	24	8
FP Bass	36	36	35	34	33	33	32	30	27	23	9
FP DLF	36	35	34	34	33	31	30	28	25	21	5
	0	10	20	30	40	50	60	70	80	90	100

G1/DB–L

	100	90	80	70	60	50	40	30	20	10	0
Spec. Sim.	28	31	33	33	34	33	33	33	32	30	28
Noisiness	33	34	33	32	31	30	29	28	26	21	8
Perc.	34	33	33	33	31	30	28	26	24	19	8
FP	33	34	33	33	33	33	31	30	29	27	24
FP Gravity	33	34	33	33	32	31	30	27	24	22	8
FP Bass	33	34	33	32	31	30	29	27	24	20	9
FP DLF	34	33	32	31	31	28	27	25	23	19	5
	0	10	20	30	40	50	60	70	80	90	100

Figure 2.25: Summary of the 4×8008 results for combining 7 similarity measures computed on DB-MS and DB-L using G30S and G1. All values are given in percent. The mixing coefficients for the spectral similarity (the first row) are given above the table, for all other rows below. For each entry in the table of all possible combinations the highest accuracy is given. For example, the second row, third column depicts the highest accuracy obtained from all possible combinations with 20% Noisiness. The not specified 80% can be any valid combination of mixing coefficients, e.g. 80% spectral similarity, or 70% spectral similarity and 10% FP Gravity etc.

Rank	Noisiness	Perc.	FP	FP Gravity	FP Bass	FP DLF	Spec. Sim.	DB-MS		DB-L		Score
								G1	G30S	G1	G30S	
1			10	10	10		70	67.4	67.4	32.4	35.2	6.14
2		10	10	10	10		60	67.1	66.4	33.0	34.6	5.83
3		10		10	10		70	66.8	66.4	31.8	34.7	5.46
4			10	10			80	67.4	65.7	32.1	34.4	5.44
5			10	10	20		60	66.1	66.9	31.5	34.9	5.42
6		10	20		10		60	65.7	66.4	32.6	34.5	5.36
7	10		10	10	10		60	63.9	66.1	$\underline{\overline{33.6}}$	$\underline{\overline{35.6}}$	5.35
8		10		10			80	66.8	66.1	31.8	34.1	5.26
9	10		20	10		10	50	64.9	66.1	32.7	35.1	5.25
10			10	10	10	10	60	67.2	66.8	30.9	33.9	5.25
11				10	10		80	$\underline{\overline{68.2}}$	66.7	31.0	32.9	5.25
25	10		20	10			60	64.1	65.2	32.7	$\underline{\overline{35.6}}$	4.92
515				30		20	50	66.0	$\underline{\overline{68.4}}$	26.5	29.5	3.15
2666							100	62.8	62.4	27.6	25.0	0.00

Table 2.10: Top 10 combinations. The last line is the baseline using only spectral similarity. The first column is the rank. All values (other than the ranks) are given in percent. Values marked with a line below and above are the highest accuracy achieved for the specific combination of similarity measure and collection.

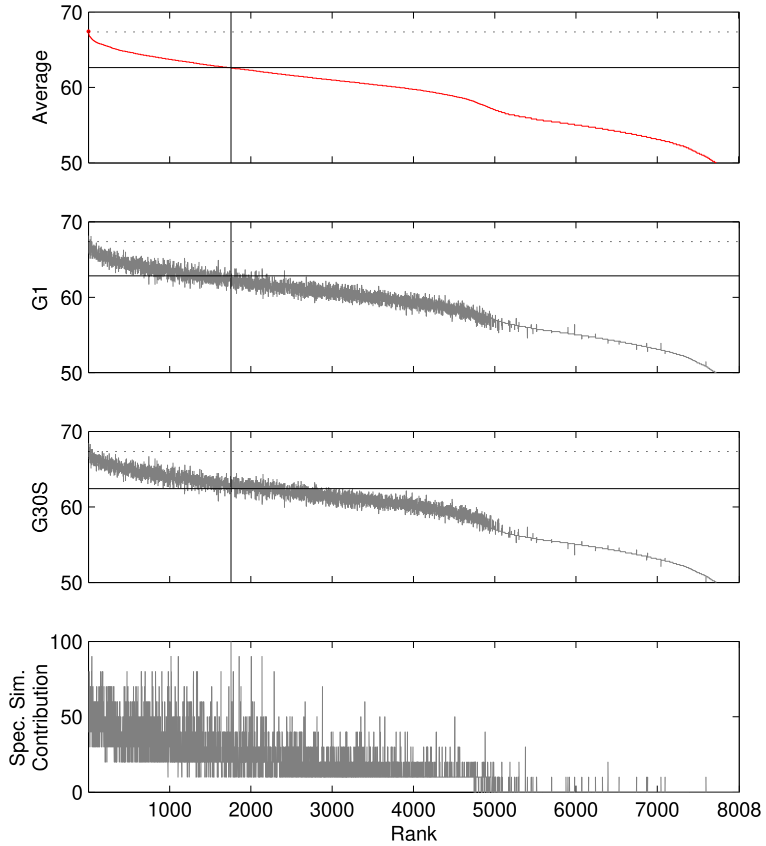


Figure 2.26: Accuracies for all 8008 combinations on DB-MS. The first plot shows the average accuracy for each combination. (That is, the average performance when using G1 and G30S as spectral similarity.) The combinations are sorted according to this average score. The dotted horizontal line marks the accuracy of the combination which scored best in average on DB-MS and DB-L (see Table 2.10). The solid horizontal line marks the accuracy of the baseline. The solid vertical line marks the position of the baseline combination (i.e., the combination which uses 100% spectral similarity). The last plot shows the contribution of the spectral similarity.

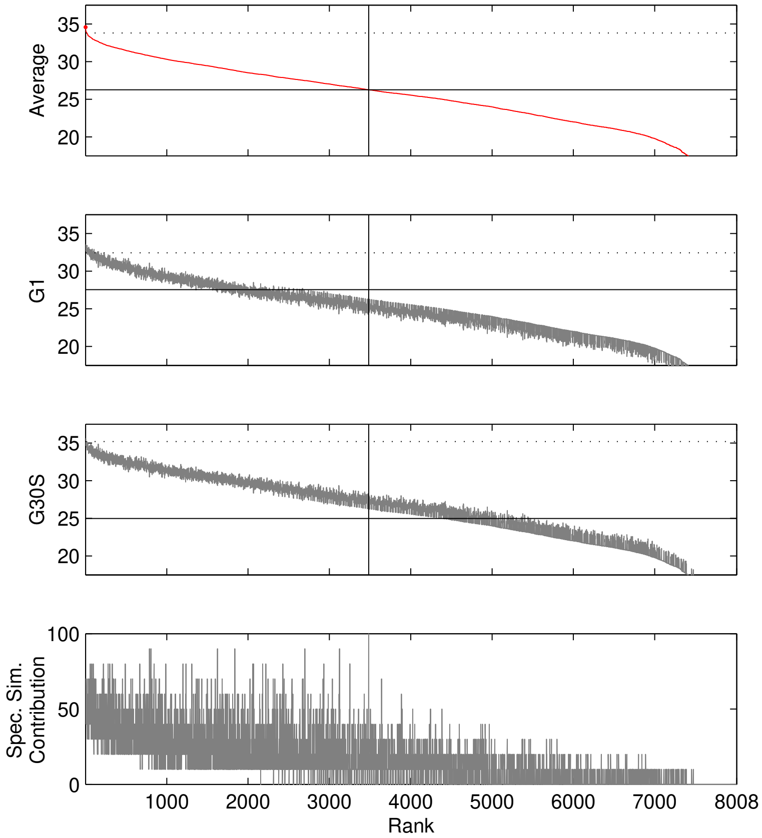


Figure 2.27: Accuracies for all 8008 combinations on DB-L. See Figure 2.26 for a description of the plots.

the baseline. On the other hand, the improvements beyond the dotted line (the combination which performed best in average) are obviously not significant.

The variances in Figure 2.28 are significantly higher compared to those in the previous two figures. In fact, except for the G30S/DB-L only the top 100 combinations seem to offer significant improvements compared to the baseline. This clearly shows the importance of using more than one music collection when evaluating similarity measures.

2.3.5.3 Overfitting

To demonstrate overfitting the DB-S collection is used. The 8008 combinations are evaluated for G1. The best combination has an accuracy of 43% (the baseline is 31%). The combination which performed best in average on DB-L and DB-MS (G1C) achieves 38%. Figure 2.29 shows that the variance is extremly high. In fact, the 11-th best combination for DB-S has a score of -0.8 percentage points in

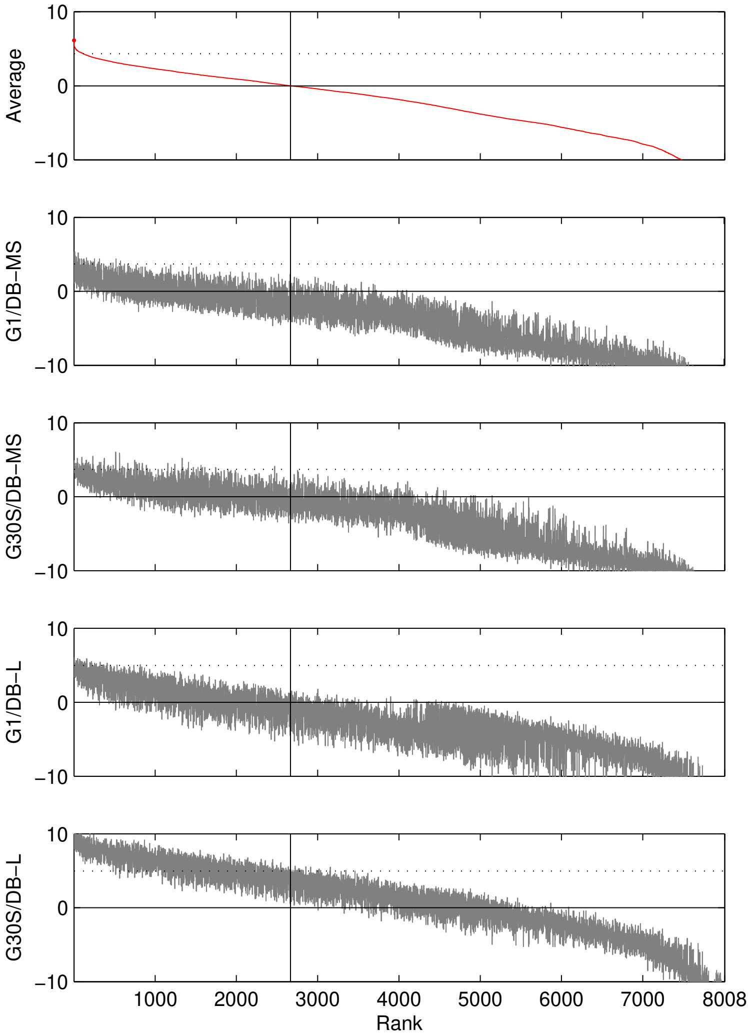


Figure 2.28: Accuracies for all 8008 combinations sorted by their average scores on DB-MS and DB-L. For a description of the plots see Figure 2.26.

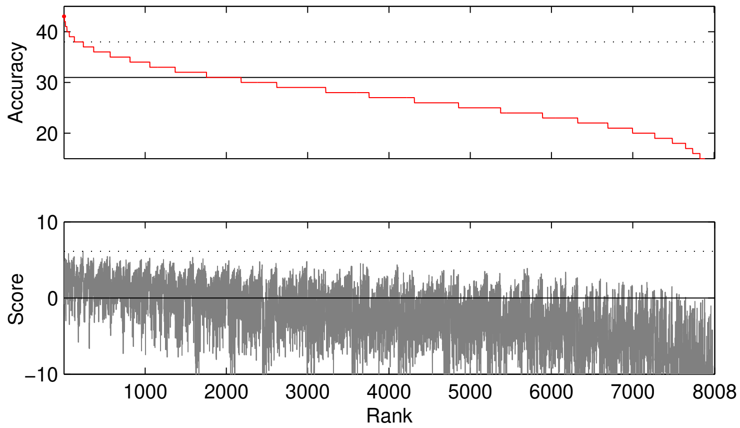


Figure 2.29: Demonstration of overfitting on the DB-S collection. The upper plot shows the (sorted) accuracies for the 8008 combinations. The lower plot shows the average score for DB-MS and DB-L. The dotted line is the score of the best combination in average (G1C). The solid line is the baseline (G1).

average on DB-L and DB-MS. The best combination on DB-S has a average score of 2.6 on DB-L and DB-MS.

There are a number of combinations in the top ranked positions which have accuracies lower than the baseline. Thus, the optimization in this case can be interpreted as classical overfitting. What has happened is that the collection used for the optimization of the combination was too small. However, DB-MS and DB-L are also quite small collections which only reflect a tiny fraction of all different styles of music. The following subsection will evaluate if the improvements are significant (to human listeners) and if they are observable on larger collections.

2.3.6 Evaluation

This subsection will first analyze the difference between the baseline and the improvements. Second, the impact of the length of the analyzed audio is studied. Finally, a listening test is presented. The measures of interest are:

- **G1**: is much faster than G30S and the combinations perform more or less identically in terms of quality.
- **G1C**: is the best combination found in Subsection 2.3.5. G1C is the combination of 70% G1 + 10% FP + 10% FP Gravity + 10% FP Bass.
- **M'05**: is the similarity measure submitted by the author to the MIREX 2005 genre classification contest [Pam05] and combines 65% G30S + 15% FP + 15% FP Gravity + 5% FP Focus. M'05 was optimized using DB-L, DB-MS, DB-S, and DB-ML.

2.3.6.1 Analyzing the Improvement

Two questions will be dealt with in the following paragraphs. First, what is the difference between G1 and G1C? Second, is this difference related to overfitting? To show the difference between G1 and G1C confusion matrices are used for DB-L (Figures 2.30-2.32) and later for DB-XL (Figures 2.34-2.36).

As can be seen in Figure 2.30 there are a number of confusions which make sense. For example, death metal is confused with heavy metal, and German hip hop with hard core rap. However, quite a number of confusions are very surprising such as heavy metal/trash with a capella. Figure 2.31 shows that this particular confusion does not occur for G1C. However, other confusions are introduced which are not much better. Figure 2.32 shows the differences between the two. Most noticeable improvements are that 30 punk piece are no longer classified as heavy metal/trash, 28 German hip hop and 19 hard core rap pieces are no longer estimated to sound like jazz. A capella and punk are better distinguished from other genres, etc.

However, on the other side, 35 folk rock pieces are classified as punk, the confusion between German hip hop and hard core rap is much larger, jazz is generally misclassified more often, etc. Overall, one would expect that fewer confusions should occur between genres which have strong differences in characteristics related to rhythm, tempo, or beats. On the other hand, genres which exhibit large variations in these aspects and less variation in instrumentation might tend to be confused more often. Although there are such tendencies it is rather difficult to find evidence for this in the confusion matrices.

The next question is: How does G1C perform on data not used for testing? In particular, is G1C a result of overfitting? That is, is G1 just as good (or even better) in general? To answer these questions the nearest neighbor accuracies (with leave-one-out cross-validation) were computed for all six collections. Of interest are the four not used to optimize the parameters (DB-S, DB-ML, DB-XL, DB-30). The results are shown in Table 2.11.

As can be seen, G1C performs better than G1 (measured using an artist filter and nearest neighbor classifier) on DB-S (5% points), DB-ML (3% points), and DB-30 (2% points). However, on the large DB-XL collection there is no improvement (when considering only the nearest neighbor). Furthermore, the improvement on DB-ML is not a big surprise as DB-MS is a subset of DB-ML. M'05 performs slightly worse than G1C on DB-ML, DB-L, DB-XL, and slightly better on the other collections. Considering the much larger number of features used in this thesis (compared to [PFW05b]) the improvements are rather disappointing. An important question is whether these improvements matter to a listener. That is, how does the improvement of 3 percentage points on genre classification accuracies relate to a listener's experience? This question will be dealt with in Subsection 2.3.6.3. (As will be shown, in the case of DB-L the 3 percentage points improvement of genre classification accuracies equals an improvement of over 7 percentage points on listeners' similarity ratings.)

	AC	AJ	Blu	BN	Cel	DM	DnB	DT	Ele	ED	FR	GH	HC	HM	Ita	Jaz	JG	MM	Pun	Reg	Tra	T2
A Cappella	23	6	1	15					2		14	10	1		9	8	21	2				
Acid Jazz		8	4					4	1	8		9	4	1	3	22			1			3
Blues	2		11		7			6				6	3		17	5			5	1		
Bossa Nova	2	10		14	5			2			1	7	3		6	10	9		1	2		
Celtic	2	1	8	6	52			1	2		10	1		2	31	2	6	3	4		1	
Death Metal						1					7			38	5	1		3	14			
DnB		1					10	5	1	13		9	3		1	3		1	1		11	12
Downtempo	1	7	1	4	2		1	8	3	2	3	21	11	2	6	15	2	3	3	5	1	16
Electronic		5	1	1	8		1	3	2	2	3	9	6	1	3	7	2	1			3	5
Euro−Dance		5			2					43		10	4		3	3		1	8		4	13
Folk−Rock	4	6	7		7	5		4		1	41	22	4	20	46	1		24	31	2	2	6
German Hip Hop	4	2	6	1				5			2	28	15	1	3	48		1	2		2	3
Hard Core Rap			2					3		1		32	44			31		1		7		1
Heavy M. – Thrash	1		1		1	15				1	18	7	1	119	5			6	65			1
Italian	3	1	17		7	2				1	11	26	5	1	30	13		4	19	2		
Jazz	1	13	3	4				3	2	1		9	3		1	17	2		2		2	
Jazz Guitar	2	6		12	6			1	1						6	9	27					
Melodic Metal		2	2		6				1	1	15	1		15	16	1		35	21		2	1
Punk	1		4		2	6		1	1	1	8	17	1	43	21			33	115	1		
Reggae			4								1	9	21		3	2				3		1
Trance	1	3				1	3	1	1	12	2	10	5		1			1	6		4	13
Trance2								1		4		4	8		2	1					1	21

Figure 2.30: Confusion matrix for DB-L and G1. Rows are the true genres, columns are the predicted genres.

	AC	AJ	Blu	BN	Cel	DM	DnB	DT	Ele	ED	FR	GH	HC	HM	Ita	Jaz	JG	MM	Pun	Reg	Tra	T2
A Cappella	57		2	4	6		1		2		3	3	2		3	5	23		1			
Acid Jazz		4	1		1		1	2		11		12	6	1	4	11			7	4	2	1
Blues	3	1	14		6			4			1	4	4		12	4	5		3	2		
Bossa Nova	2	3	2	11	5			4	2		4	6			11	5	14	1	2			
Celtic	5	2	4	13	50			2			8	4			24	1	3	9	6		1	
Death Metal						2					6			38	2	1		2	18			
DnB		3	2	2			12	4	2	1	7	15	6	1	1	3		1	6		4	1
Downtempo		3	5	4	2	1	5	8	2	1	10	31	4		15	10	1		12	1	2	
Electronic		7	2	2	2	1		3		13	2	3	6	1	5	4	4	2	1		2	3
Euro−Dance		4							2	47		9	2		4	2		1	1	2	12	10
Folk−Rock	3	1	1		5	5	2	5		3	47	1	3	20	42	2	1	20	66	1	5	
German Hip Hop	2	8	1				1	7		1	1	32	37	1	5	20		1	4	1		1
Hard Core Rap	1	1						1			1	58	37			12				11		
Heavy M. – Thrash			1		2	18	1	1			34	3		118	1			17	45			
Italian	1	4	8	3	11					2	34	15	2		39	3		1	19			
Jazz	1	5	5	6			1	7	1	2	2	16	2		2	5	6		1	1		
Jazz Guitar	1	2	3	19	4				1			1			2	6	31					
Melodic Metal		1			2	3		1			8	2		20	4		1	45	32			
Punk	1	1	3		2	8		1			24	4	1	13	13	1		21	161		1	
Reggae		1	1							1	1	7	20			1				11	1	
Trance							1	1		16	4	8	3	1		2		5	2	3	17	1
Trance2		4			1				1	7					2						3	24

Figure 2.31: Confusion matrix for DB-L and G1C.

	AC	AJ	Blu	BN	Cel	DM	DnB	DT	Ele	ED	FR	GH	HC	HM	Ita	Jaz	JG	MM	Pun	Reg	Tra	T2
A Cappella	34	−6	1	−11	6		1				−11	−7	1		−6	−3	2	−2	1			
Acid Jazz		−4	−3		1		1	−2	−1	3		3	2		1	−11			6	4	2	−2
Blues	1	1	3		−1			−2			1	−2	1		−5	−1	5		−2	1		
Bossa Nova		−7	2	−3				2	2		3	−1	−3		5	−5	5	1	1	−2		
Celtic	3	1	−4	7	−2			1	−2		−2	3		−2	−7	−1	−3	6	2			
Death Metal						1					−1				−3			−1	4			
DnB		2	2	2			2	−1	1	−12	7	6	3	1					5		−7	−11
Downtempo	−1	−4	4			1	4		−1	−1	7	10	−7	−2	9	−5	−1	−3	9	−4	1	−16
Electronic		2	1	1	−6	1	−1		−2	11	−1	−6			2	−3	2	1	1		−1	−2
Euro−Dance		−1			−2				2	4		−1	−2		1	−1			−7	2	8	−3
Folk−Rock	−1	−5	−6		−2		2	1		2	6	−21	−1		−4	1	1	−4	35	−1	3	−6
German Hip Hop	−2	6	−5	−1			1	2		1	−1	4	22		2	−28			2	1	−2	−2
Hard Core Rap	1	1	−2					−2		−1	1	26	−7			−19		−1		4		−1
Heavy M. – Thrash	−1				1	3	1	1		−1	16	−4	−1	−1	−4			11	−20			−1
Italian	−2	3	−9	3	4	−2				1	23	−11	−3	−1	9	−10		−3		−2		
Jazz		−8	2	2			1	4	−1	1	2	7	−1		1	−12	4		−1	1	−2	
Jazz Guitar	−1	−4	3	7	−2			−1				1			−4	−3	4					
Melodic Metal		−1	−2		−4	3		1	−1	−1	−7	1		5	−12	−1	1	10	11		−2	−1
Punk		1	−1			2			−1	−1	16	−13		−30	−8	1		−12	46	−1	1	
Reggae		1	−3							1		−2	−1		−3	−1				8	1	−1
Trance	−1	−3				−1	−2		−1	4	2	−2	−2	1	−1	2		4	−4	3	13	−12
Trance2		4			1			−1	1	3		−4	−8			−1					2	3

Figure 2.32: Difference between the DB-L confusion matrix between G1C and G1 (G1C minus G1).

		DB-S	DB-MS				DB-ML				DB-L			
	AF	1	1	5	10	20	1	5	10	20	1	5	10	20
G1	–	56	78	76	74	70	81	79	77	74	76	73	72	69
	+	31	63	65	66		54	57	59		28	30	31	31
G1C	–	60	80	78	75	74	80	79	78	76	70	70	69	65
	+	36	67	68	68		57	59	60		32	35	34	33
M’05	–	56	82	76	72	70	80	79	77	74	74	72	70	67
	+	38	67	67	64		55	56	60		30	33	34	35

		DB-XL				DB-30		
	AF	1	5	10	20	1	5	10
G1	–	59	58	58	57	41	36	34
	+	31	33	35	37	11	11	13
G1C	–	55	55	55	54	39	34	32
	+	31	35	36	38	13	13	15
M’05	–	55	55	55	54	41	36	35
	+	30	33	35	36	14	13	14

Table 2.11: Leave-one-out cross-validation results using a k-NN classifier with $k = 1, 5, 10, 20$. If in any of the genres less than k items remained after applying the artist filter, then the k value was not evaluated and left blank.

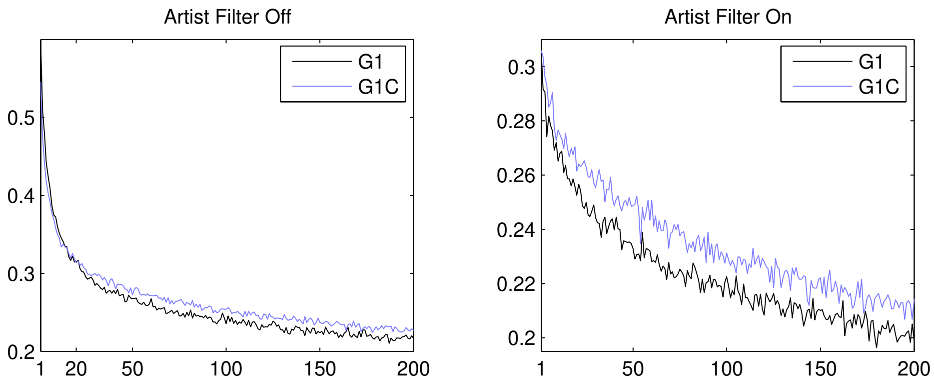


Figure 2.33: Detailed differences between G1 and G1C measured on DB-XL without and with artist filter. The plots show the average percentage of pieces at a given rank position (the ranks are sorted according to the distance to the seed) which belong to the same genre as the seed. In particular, the closest 200 pieces to each piece in DB-XL are analyzed.

Figure 2.33 analyzes the difference between G1 and G1C on DB-XL in more detail. As can be seen, both perform the same for the nearest neighbor. However, if more neighbors are considered, then G1C consistently outperforms G1. A question which is not answered in this thesis is whether these improvements are noticeable by a human listener (within the context of an application).

Interesting in Figure 2.33 is that the artist filter effect is very noticeable. In particular, if no artist filter is used, the performance for the 20 nearest neighbors is very high. After these, the performance is very similar to the one of using an artist filter. In average there are 27 pieces per artist. This indicates that for most pieces 2/3 of the other pieces by the same artist are spectrally extremely similar.

Figures 2.34-2.36 show the confusion matrices for G1 and G1C on DB-XL. Both completely fail to classify Christmas songs (which is not a surprise) and easy listening. G1 frequently and G1C very frequently confuse country, alternative & punk, and rock. G1C distinguishes electronica/dance from other genres better than G1. In particular, the confusion with hip hop/rap is reduced. Furthermore, classical music is better distinguished by G1C. On the other hand, jazz is better distinguished by G1.

Another question that has arisen from these results is why the performance of DB-30 is so extremely low. One possible answer might be that the similarity measures do not work for very short (30 second) excerpts. This is analyzed in the following paragraph.

	A&P	Blu	Chr	Cla	Cou	EaL	E/D	Fol	HHR	Jaz	Lat	NeA	Pop	R&B	Reg	Roc	Wor
Alternative & Punk	430	71	6	10	399	18	113	20	101	172	64	22	138	254	10	227	21
Blues	47	11	1		34	3	4	2	13	45	3		24	34	5	24	4
Christmas	8			8	4				3	27	4			9		4	1
Classical	17		16	391	9		5	1	3	118	7	12	10	27		8	1
Country	366	51	1	12	411	15	87	12	98	159	60	14	143	163	7	333	36
Easy Listening	25	2			19		6	1	4	28	5	2	2	12		9	4
Electronica/Dance	245	39	2	6	230	5	410	5	179	245	32	10	95	181	19	147	10
Folk	26	4		1	19		5	34	15	19	1		10	10		12	2
Hip Hop/Rap	65	11		4	48	2	31	9	347	23	6		17	88	7	22	2
Jazz	217	46	13	110	152	13	70	10	35	2106	75	30	39	169	8	209	27
Latin	70	11		5	44	4	12	4	19	67	17	1	24	53	2	27	6
New Age	43	3	1	19	34	1	14		2	63	10	8	7	29	2	21	6
Pop	117	8		3	115	6	33	7	14	32	20	4	54	72	2	60	4
R&B	235	44	1	4	108	10	34	11	81	125	47	6	77	218	7	74	15
Reggae	15	5		1	16	1	18		19	11	2	2	4	21	3	13	
Rock	270	38	1	10	310	5	46	13	65	162	36	6	107	127	7	258	30
World	59	13	1	5	32	3	15		6	43	11	4	17	43		36	6

Figure 2.34: Confusion matrix for DB-XL and G1.

	A&P	Blu	Chr	Cla	Cou	EaL	E/D	Fol	HHR	Jaz	Lat	NeA	Pop	R&B	Reg	Roc	Wor
Alternative & Punk	419	46	7	13	406	28	80	18	94	178	58	23	155	226	6	294	25
Blues	60	3	1		51	5	5	4	7	42	4	1	9	33		25	4
Christmas	10	3		10	5	1	2			27	1			3		6	
Classical	22	1	21	428	5		6		1	86	5	11	5	26		8	
Country	426	50	4	18	460	11	71	15	70	115	56	16	138	158	2	333	25
Easy Listening	24	8			18		7		5	23	5		4	14		10	1
Electronica/Dance	257	29	6	17	200	10	469	8	117	224	53	24	91	173	22	143	17
Folk	18	1	2	2	27	1	4	31	7	19	1		13	13		18	1
Hip Hop/Rap	75	13	1	2	42	1	33	7	335	23	9	1	12	99	9	18	2
Jazz	244	56	16	115	167	20	67	11	45	2039	80	32	37	178	4	184	34
Latin	68	4		7	49	2	22		23	60	24	2	21	40	1	33	10
New Age	46	1	1	26	41	1	17			52	8	11	7	12		32	8
Pop	129	11		4	130	3	35	4	6	41	19	3	49	56	1	53	7
R&B	235	56		5	134	17	47	8	86	110	38	5	65	208	6	71	6
Reggae	15	3	1		12		15	1	30	18	5	1	2	22	1	5	
Rock	291	31	4	18	355	6	42	6	34	146	40	14	116	128	6	228	26
World	43	5	3	8	44	5	20	2	3	42	14	10	14	30	2	45	4

Figure 2.35: Confusion matrix for DB-XL and G1C.

	A&P	Blu	Chr	Cla	Cou	EaL	E/D	Fol	HHR	Jaz	Lat	NeA	Pop	R&B	Reg	Roc	Wor
Alternative & Punk	–11	–25	1	3	7	10	–33	–2	–7	6	–6	1	17	–28	–4	67	4
Blues	13	–8			17	2	1	2	–6	–3	1	1	–15	–1	–5	1	
Christmas	2	3		2	1	1	2		–3		–3			–6		2	–1
Classical	5	1	5	37	–4		1	–1	–2	–32	–2	–1	–5	–1			–1
Country	60	–1	3	6	49	–4	–16	3	–28	–44	–4	2	–5	–5	–5		–11
Easy Listening	–1	6			–1		1	–1	1	–5		–2	2	2		1	–3
Electronica/Dance	12	–10	4	11	–30	5	59	3	–62	–21	21	14	–4	–8	3	–4	7
Folk	–8	–3	2	1	8	1	–1	–3	–8				3	3		6	–1
Hip Hop/Rap	10	2	1	–2	–6	–1	2	–2	–12		3	1	–5	11	2	–4	
Jazz	27	10	3	5	15	7	–3	1	10	–67	5	2	–2	9	–4	–25	7
Latin	–2	–7		2	5	–2	10	–4	4	–7	7	1	–3	–13	–1	6	4
New Age	3	–2		7	7		3		–2	–11	–2	3		–17	–2	11	2
Pop	12	3		1	15	–3	2	–3	–8	9	–1	–1	–5	–16	–1	–7	3
R&B		12	–1	1	26	7	13	–3	5	–15	–9	–1	–12	–10	–1	–3	–9
Reggae		–2	1	–1	–4	–1	–3	1	11	7	3	–1	–2	1	–2	–8	
Rock	21	–7	3	8	45	1	–4	–7	–31	–16	4	8	9	1	–1	–30	–4
World	–16	–8	2	3	12	2	5	2	–3	–1	3	6	–3	–13	2	9	–2

Figure 2.36: Difference between the DB-XL confusion matrix between G1C and G1 (G1C minus G1).

2.3.6.2 Impact of the Length of the Analyzed Audio

As mentioned in Subsection 2.2.2.5 two minutes from the center of each piece are used for analysis. Table 2.12 shows the results when using 3, 2, 1 or 0.5 minutes of the center of each piece for analysis (using DB-MS). The general tendency is that the longer the analyzed excerpt, the better the results. However, the difference between 120 and 240 is only minimal, while the feature extraction time is twice as long. Thus, the answer to the question raised above is that the performance on DB-30 is so low due to the type of music and the taxonomy. The classification accuracies below 15% clearly point out the limits of using audio-based similarity measures to automatically classify music into the taxonomoies used by music distributors.

2.3.6.3 Listening Test

So far, all improvements have been measured using genre classification accuracies. However, it remains unclear if these percentage points relate to judgments by human listeners. To measure the relationship the following listening test was conducted.

Setup

The listeners were presented a seed song and two similar pieces. One of these pieces is the nearest neighbor according to G1, the other according to G1C. The listeners are not informed which of the two alternatives is which. Furthermore,

	G1							G1C						
Length	AF −				AF +			AF −				AF +		
Seconds	1	5	10	20	1	5	10	1	5	10	20	1	5	10
30	74	73	70	68	59	65	62	75	74	73	69	63	65	65
60	79	74	71	69	63	65	64	77	74	73	71	64	66	65
90	80	76	73	69	63	66	66	79	77	75	72	65	68	66
120	78	76	74	70	63	65	66	80	78	75	74	67	68	68
180	81	77	76	70	64	66	67	82	77	77	75	68	68	69
240	81	76	76	72	63	67	68	81	79	77	73	66	69	69

Table 2.12: Impact of the length of the analyzed section of each piece on the classification accuracies. All accuracies are given in percent and are computed on DB-MS.

all artist, album, and track names are removed. The listeners were asked to rate: how much they like the seed, and how similar they consider each of the two songs. The scale used ranges from 1 to 9. The values are defined as follows: 1 terrible, 5 acceptable, 9 could not be better.

A stratified random sample of 100 seeds was selected from DB-L. Stratified, such that the ratio of seeds for which G1 and G1C have a nearest neighbor from the same genre as the seed corresponds to the distribution in the data. Thus for G1C slightly more seeds were selected which had a nearest neighbor from the same genre. Ensuring a fair distribution is important if one assumes that there is a correlation between pieces being from the same genre and pieces being similar. Furthermore, a total of 11% of all possible seeds were ignored because for those G1 and G1C have the exact same nearest neighbor. That is, in 11% of the cases there is no difference at all (if only the nearest neighbor is considered). The reported results only apply to the remaining 89% of the cases. This needs to be considered when computing the overall difference.

One could argue that 100 seeds is not enough to reflect on one side the variation of music in the collection, and on the other side the variation of the performance of each similarity measure. In fact, it would be possible to select 100 seeds where one measure performs perfect, or even easier to select 100 seeds where one measure fails. Furthermore, in addition to using as many seeds as possible, it would be very interesting to analyze more than just the nearest neighbor. Ideally, the listeners would be asked to rate the closet 20 or more pieces. (As shown in Figure 2.33 sometimes the differences are only measurable beyond the nearest neighbor.) However, organizing a large scale listening test was beyond the scope of this thesis. To show significant differences between G1 and G1C on DB-XL a very large scale test would have been necessary. Instead the listening test reported here uses DB-L where the difference in classification accuracies is 3 percentage points.

The users were asked not to focus on melody, harmony, cultural context, or rhythm but focus on overall sound characteristics. In general the listeners who

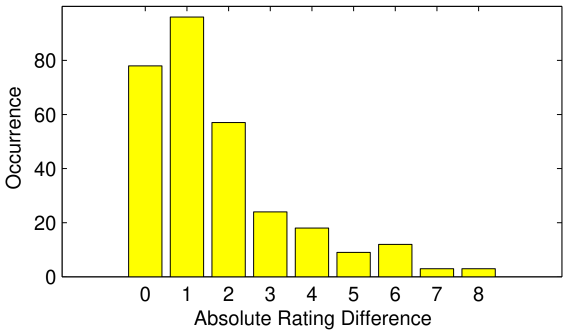


Figure 2.37: Histogram of the absolute difference of differences for listeners' ratings. For example, listener A rates the similarity of the two songs with 5 and 7 respectively, and listener B rates the similarity with 7 and 8. Then the absolute difference between the differences is 1 point ($|(5-7)-(7-8)|$).

were musically untrained had less problems with this definition of similarity. The 100 seeds were randomly grouped into 10 blocks. Most listeners were only asked to rate one block and only few rated 2 blocks. The average time to answer one block was between 15–40 minutes.

The listeners were allowed to listen to each songs as long, and as many times, and in any sequence as they wanted. They were told that they could use the position slider of the media player to zap through the piece (e.g. to skip the introduction). In particular, the listeners were asked to get an impression of the piece but not listen to the entire piece. The total number of participants was 25 of which 11 had musical training. There were 4 female and 21 male participants. The 2 neighbors to each seed were rated by 3 listeners resulting in a total of 600 similarity ratings.

Results

The correlation between the listeners' ratings was surprisingly high. This is shown in Figure 2.37. As can be seen, in many cases the listeners fully agreed on the difference between the differences of the two nearest neighbors. In most cases, the difference was 1 point. Only in a few cases the listeners truly disagreed. This high correlation between listeners albeit the weak definition of similarity confirms the observations in [LS01].

The overall mean rating for G1 on DB-L is 5.73, and for G1C it is 6.37. The absolute difference is 0.64 which is less than the resolution of the scale. This corresponds to 8% points on the scale from 1 to 9. Considering that in 11% of the cases the nearest neighbors for G1 and G1C are identical (and have been ignored in the test) the overall improvement is about 7 percentage points or about 0.57 points on the scale. This small difference makes it obvious that the even smaller difference

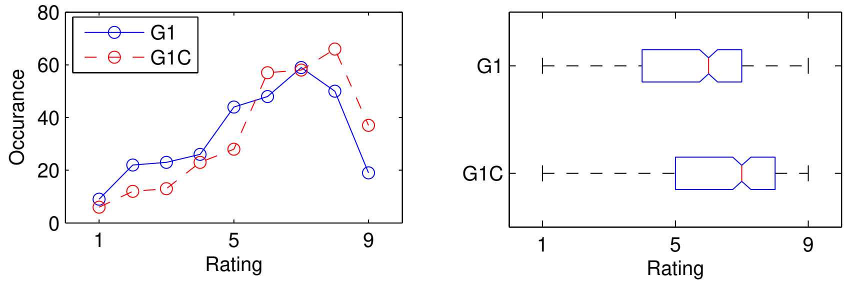


Figure 2.38: Histogram and box plots of all ratings comparing G1 to G1C.

measured on DB-XL could not be analyzed with such a small scale listening test.

Nevertheless, there are some very significant differences. For example, for G1C 37 times a match was rated with 9 (i.e. "perfect"), while for G1 only 19 times a match was rated with the highest. A histogram of all ratings is shown in Figure 2.38.

No nearest neighbor got consistently the lowest rating (i.e. "terrible") from the three listeners it was exposed to. However, some pieces got very low scores. The following four examples demonstrate the weaknesses of the similarity measures (2 examples are from G1C, 2 from G1).

A. The seed song is from DJ Shadow and is titled "What does your soul look like (Part 1 - Blue Sky Revisit)". It can be described as calm electronic music. The nearest neighbor computed with G1C is "Amazing Discoveries" by Fettes Brot. The users rated this match with 1.67 in average. This match is actually not music. Instead, German speakers simulate a television advertisements show. Perhaps such mistakes could be avoided using metadata. Nevertheless, it is very difficult to understand why the similarity measure considers the two tracks similar. The nearest neighbor for G1 is "Quoterman" by Kinderzimmer Productions and is rated with 5 in average. It is a German rap song.

B. The seed song is "Run Along for a Long Time" from the Heavenly Light Quartet, which is an a capella gospel group. The song has a few rap elements. G1C matches the rap song "Constant Elevation" by the Gravediggaz to it, which is rated with 1.67 in average. While this match is understandable from its acoustic aspects it is a complete failure from a cultural point of view. Again, metadata could perhaps be used to filter such mistakes. G1 matches a song by the South African a capella group named Ladysmith Black Mambazo. The song is titled "Liph' lquiniso". This match is far better and is rated with 7 in average.

C. The seed is FantasMic by Nightwish. The piece can best be described as gothic, romantic metal ballad. It starts relatively soft, with a calm female voice, and electrical guitars with drums. The second half is a lot more energetic and faster (the song is longer than 8 minutes). G1 matches “Pay Attention” by US3 to it which is rated in average with 2. The song features female rappers and very different instruments, for example, it contains no electrical guitars. On the other hand, G1C matches Luminous by Stratovarius which is rated with 4 in average. Luminous is a heavy metal ballad and not only culturally much similar.

D. The seed is “Land of ...” by St. Germain. It can be described as fusion of jazz and house music. There are no vocals. G1 matches “Michi Gegen Die Gesellschaft” by Die Fantastischen Vier. This German hip hop song features a male rapper and has a very different style compared to seed. It is rated with 2 in average. G1C also matches a song from Die Fantastischen Vier: “Weiter Als Du denkst” which is rated with 4.67 in average. It features male and female voices, is slower and much softer than other match.

In total 32 times a G1 match was considered unacceptable (i.e, the ratings were in average <5), and 14 times for G1C. There were 9 songs which were consistently (by all three listeners) rated smaller than 5, and 6 songs for G1C. For some applications even a low number of such failures can be unacceptable.

One of the main purposes of this listening test was to see how the results from the genre-based evaluation correspond to human listeners’ ratings. For one, this has already been supported by the fact that G1C performs better than G1 in the listening test. To further analyze this, Figure 2.39 shows the ratings for pieces from the same genre as the seed, and the ratings for pieces from different genres as the seed. One would expect that ratings are significantly higher for pieces from the same genre. The average rating for songs from the same genre is 6.21, the average rating for songs from a different genre is 5.89. The difference is smaller than expected. However, it is necessary to consider that according to the similarity measure all matches are very similar to the seeds. If additional pieces with large distances (according to the similarity measure) had been included the numbers would have been quite different. Although these results confirm the assumption that genre classification can be used to evaluate similarity measures they also demonstrate the limits of this approach. In particular, more songs from different genres are rated as “perfect” matches than songs from the same genre.

Finally, it is interesting to study if there are any correlations between how much the listeners liked a seed and how the nearest neighbors were rated. For example, if listeners do not like heavy metal (“it all sounds the same”), they might tend to not differentiate as much as if they like the music. The correlation between how much the users like the seeds and the absolute difference in ratings between the two matches is 0.05. There is a positive tendency, but it is neglectable. A second

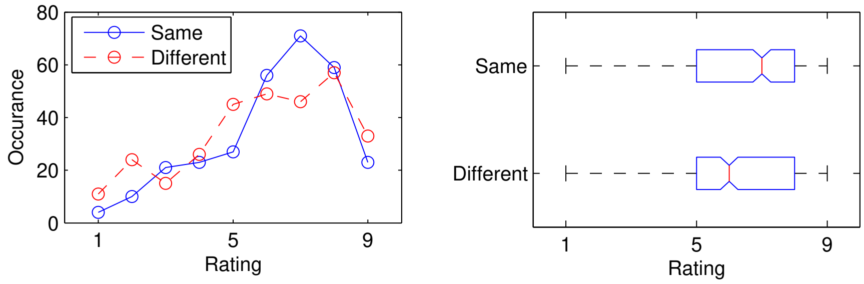


Figure 2.39: Histogram and box plots of ratings for pieces from the same genre as the seed compared to ratings for pieces from different genres as the seed.

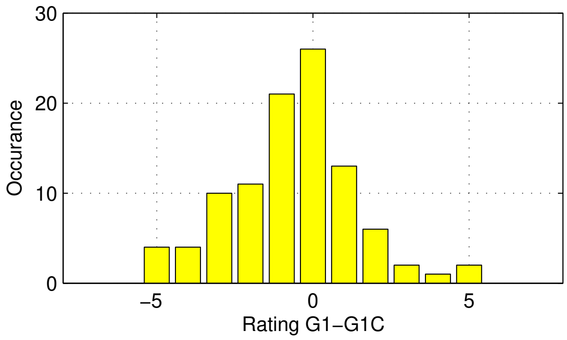


Figure 2.40: Histogram of the average difference between ratings per seed (G1 - G1C). The shift to the left is significant according to the Wilcoxon test.

question is if listeners will generally tend to give the similarity measures lower ratings if they dislike the seed song. The correlation between how much they like the seed and how high they rated the neighbors is 0.13. Again, this is a positive tendency, but it seems neglectable.

Statistical Significance

To test the statistical significance of the difference between G1 and G1C on DB-L the non parametric Wilcoxon test can be used. The only assumption this test makes is that the values are distributed symmetrically around a common mean. This is confirmed to some extent in Figure 2.40. The figure shows the histogram of differences between G1 and G1C for each seed song. In particular, for each seed song all three differences (between the G1 and G1C match) are computed and the average thereof is used. Thus, the three ratings for each seed-neighbor pair are treated as multiple measurements of the same information (which are not independent). The left-sided P-value for the test is 0.0005. This means that G1C is in average rated significantly higher than G1 on DB-L.

2.4 Limitations and Outlook

The similarity ratings of the computation models are far from perfect. For example, in the listening test, users rated the average performance around 6 on a scale from 1 (terrible) over 5 (acceptable) to 9 (perfect). A interesting question is how much the similarity measures can be further improved. Aucouturier and Pachet [AP04a] observed in their experiments that it seems impossible to significantly improve spectral similarity without higher level analysis of the music. The results in this chapter confirm this. Using only low-level audio statistics there is a clear limit and it is unlikely that variations of the features discussed in this chapter will significantly improve the quality. In fact, since the publication of [AP04a] there have been only very small improvements in terms of the quality.

While it seems that the limit for low-level audio statistics has been reached (or at least is very close), there are still many opportunities based on higher level analysis. In particular, there are two directions which seem promising: harmonic analysis (including tonality, key, and chord progressions) and rhythmic analysis (including rhythm patterns and tempo). Examples of recent work on harmony include [GB05; BP05; MKC05]. Examples of recent work on rhythm include [DPW03; DGW04; GDPW04; Gou05].

However, despite these opportunities, it is necessary to realize that in the foreseeable future computational models of audio-based music similarity will not be able to judge similarity the same way humans do. Although the judgment of music similarity seems so effortless to a human listener it is a very complex process. On the other hand, certain applications might not require perfect similarity measures. For example, a certain degree of “variance” might be acceptable for automatically generated playlists.

Another important limitation of the audio-based similarity measures is that they cannot consider the quality of a piece. For example, they cannot distinguish between a novice playing around on a piano, or a famous pianist playing a Mozart sonata. Some of the difficulties of developing algorithms which are capable of rating music were pointed out in [WE04]. Furthermore, in general emotions are not considered. A few exceptions exist, for example, aggressiveness is often correlated with loudness, strong beats, and noise.

While only small improvements have been made in the recent years in terms of quality there has been a large reduction (by several factors) in the necessary computation time. Further improvements are likely. Directions which have not been investigated thoroughly include, for example, the application of indexing methods such as M-Trees [CPZ97], or multi-resolution approaches such as the one suggested in [RAPB05].

Another option to further improve the performance of audio-based similarity measures is to combine them with additional information. Such information could be based on collaborative filtering data, or web-based community data as described in the next section.

2.5 Alternative: Web-based Similarity

This section briefly describes how data available on the web can be used to compute the similarity of artists. In particular, common web pages are used. Of particular interest are pages containing artist reviews. To find these pages Google is used. Word occurrence lists are created for the retrieved pages, and artist are compared based on their word lists. In contrast to audio-based similarity measures such an approach can describe the sociocultural context of an artist. This approach is used in Chapter 3 to hierarchically organize and describe music collections at the artist level.

2.5.1 Related Work

One of the first approaches using web-based data to compute artist similarities was presented in [PWL01]. Co-occurrences on playlists from radio stations and compilation CD databases were used to cluster a set of 12 songs and a set of 100 artists hierarchically according to similarity. The approach was demonstrated using single-linkage agglomerative clustering.

Another source are common web pages [WL02; BH03]. The main idea is to retrieve top ranked sites from Google queries and apply standard text-processing techniques. Using the obtained word lists, the artist similarities are computed. A drastically simplified approach is to use the number of pages found by Google for a query containing two artist names [ZF04; SKW05b]. As the evaluation of artist similarity is quite difficult [EWBL02] it is tempting to resort to a genre classification scenario [KPW04]. Other web-based sources include expert opinions (such as those available from the All Music Guide[27]), album reviews [WE04], or song lyrics [LKM04].

A combination of audio-based and web-based sources is very desirable; however, that is beyond the scope of this thesis. First approaches demonstrating the advantages of the combination can be found, for example, in [WS02; WE04; BPS04]. A particularly interesting approach is the work presented in [CRH05] which analyzes online user profiles, news related to music, and the audio content to give recommendations. A major difficulty is that the evaluation requires music collections with a significant amount of artists.

2.5.2 Similarity Computations (Technique)

This subsection describes a technique to compute the similarity between artists. The reason for not computing similarities between tracks or albums is that one can expect more web pages describing and reviewing an artist than pages reviewing an individual song (or album). This approach is a simplified version of the approach originally presented in [WL02] and is based on standard text information retrieval

[27] http://www.allmusic.com

techniques. In [KPW04] this approach was successfully applied to classify artists into genres.

For each artist a query string consisting of the artist's name as an exact phrase extended by the keywords *+music +review* is sent to Google using Google's SOAP interface. This service is offered free of charge but is limited to 1000 queries a day per registered user. Each query returns 10 pages.[28] For each artist the 50 top ranked pages are retrieved. All HTML markup tags are removed, taking only the plain text content into account. Very frequent and unwanted terms are removed using a stop word list.[29]

Let the term frequency tf_{ta} be the number of occurrences (frequency) of word (term) t in the pages retrieved for artist a. Let the document frequency df_{ta} be the number of web pages (documents) for a in which t occurs at least once, and let $df_t = \sum_a df_{ta}$.

First, for computational reasons, for each artist all terms for which $df_{ta} < 3$ are removed. (The highest possible value for df_{ta} is 50). Then all individual term lists are merged into one global list. From this list all terms for which there is no $df_{ta} \geq 10$ are removed. (That is, for at least one artist the term must occur in at least 10 pages.)

In an experiment with 224 artists 4139 terms remained [PFW05a]. A list of the artists is available online.[30] The data is inherently extremely high-dimensional as the first 200 eigenvalues (using an eigenvalue decomposition, also known as PCA) are needed to describe 95% of the variance.

The frequency lists are combined using the term frequency × inverse document frequency ($tf \times idf$) function (here the *ltc* variant [SB88] is used). The term weight per artist is computed as,

$$w_{ta} = \begin{cases} (1 + \log_2 tf_{ta}) \log_2 \frac{N}{df_t}, & \text{if } tf_{ta} > 0, \\ 0, & \text{otherwise,} \end{cases} \tag{2.43}$$

where N is the total number of pages retrieved.

This results in a vector of term weights for each artist. The weights are normalized such that the length of the vector equals 1 (Cosine normalization) to remove the influence the document length would otherwise have. Finally, the distance between two artists is computed as the Euclidean distance of the normalized term weight vectors.

The evaluation of this approach within a genre classification context can be found in [KPW04]. For the set of 224 artists (manually assigned to 14 genres) which are used in the experiments described in Section 3.3 the classification accuracy is of 85% for leave-one-out evaluation using a nearest neighbor classifier.

[28] http://www.google.com/apis
[29] http://www.ofai.at/~elias.pampalk/wa/stopwords.txt
[30] http://www.cp.jku.at/people/knees/artistlist224.html

2.5.3 Limitations

One of the main problems is that this approach relies on artist names. In many cases this name might have several meanings, making it difficult to retrieve relevant web pages. Whitman and Lawrence [WL02] pointed out some problematic examples such as “Texas” or “Cure”. Another problem is that many new and not so well known artist do not appear on web pages. Furthermore, it is unclear how well this technique performs with artists from other (non-western) cultures.

2.6 Conclusions

In this chapter, an introduction to computational models of audio-based music similarity was given. A number of different approaches were described. A procedure to select the most promising features for combinations was demonstrated. In particular, the number of candidates was reduced from 14 down to 6. This reduced the number of combinations in the parameter space from over 2 million to 8008.

Furthermore, a procedure to optimize the weights for a linear combination of similarity measures was demonstrated using 2 collections and 2 spectral similarity measures to avoid overfitting. The importance of using an artist filter was illustrated. The optimizations were evaluated using 4 different collections. On 5 of the 6 collections used for the experiments the improvements were significant. On the largest collection (DB-XL) the optimization showed only minimal improvements compared to the baseline. Overall, the improvements were in the range of 0–5 percentage points.

An additional argument for the improvements is the reduced occurrence of anomalies. In particular, mixing features from a vector space with the spectral similarity reduced the number of pieces in the collection which are either always similar or dissimilar (to all other pieces), and increased the number of cases where the triangular inequality holds.

A listening test was presented which shows that human listeners’ similarity judgments correspond to the genre-based evaluation. This justifies efficient optimizations based solely on genre data. The listening test compared the baseline (G1) to the best (G1C) combination of similarity measures. The test showed that a 3 percentage point increase for genre classification accuracies corresponded to about 7 percentage points higher similarity ratings by human listeners.

The limitations of audio-based similarity measures were discussed. In particular, it was pointed out that there have been only small improvements in the last years in terms of quality. (While the necessary computation times have been reduced immensely.) Reaching beyond the current limits might be possible through higher level features based on harmonic or rhythmic information. Recent work in this direction seems promising. Finally, an alternative to audio-based similarity computations was presented which is based on analysis of web pages and the use of Google.

2.6.1 Optimization and Evaluation Procedures

An important part of this chapter has been the procedures to optimize and evaluate similarity measures. The three most important points have been:

1. Demonstrating the connection between human listeners' similarity ratings and the use of genre classification to evaluate similarity measures.
2. The analysis of more than just computation times and classification accuracies. In particular, anomalies (always similar, triangular inequality, and always dissimilar) were studied.
3. The use of three strategies to avoid overfitting: (3a) the use of an artist filter, (3b) using several collections for optimization and several different collections for evaluation, (3c) using different implementations of the same approach (different types of spectral similarity) to avoid overfitting (when combining spectral similarity with additional features).

2.6.2 Recommendations

Based on the observations from the experiments reported in this thesis the following recommendations can be made. For small projects where the simplicity of the code is a main issue (e.g. for a simple prototype demonstrating playlist generation) the use of G1 is recommendable. For larger projects G1C (G1 combined according to Equation 2.42) is preferable. The additional features lead to only a minor increase in computation time. On the collections used for the experiments in this thesis the quality is at least as good if not significantly better than the performance of G1. In addition, the effects of the similarity space anomalies are reduced.

For applications where the anomalies can cause serious trouble, or applications where a vector space is highly desirable, combinations where the contribution of G1 is reduced might be of interest. For example, the combination ranked 9th in Table 2.10) uses only 50% G1. Although not further analyzed in this thesis, the results presented in Figure 2.25 suggest that combinations without spectral similarity exist which achieve similar performances.

For real world applications it is necessary to use additional information to filter results (e.g. Christmas songs). One approach could be to use the technique described in Section 2.5 or any other source (including lyrics, metadata supplied by the music distributor or third parties such as Gracenote or All Music Guide, and collaborative filtering).

Chapter 3
Applications

This chapter describes three applications which are based on similarity measures. The first application (Section 3.2) is the organization and visualization of music collections using a metaphor of geographic maps [Pam01; PRM02b; PRM02a; Pam03; PDW03a; PGW03; PDW04]. The second application (Section 3.3) is the organization of music collections at the artist level. In contrast to the first application a fuzzy hierarchy is used to structure the collection and automatic summaries are created based on words found on web pages containing the artist's name [PHH04; PFW05a]. The third application (Section 3.4) is the dynamic generation of playlists. Feedback given by the users by pushing a skip button is used to dynamically improve a playlist [PPW05a].

3.1 Introduction

There are various applications for music similarity measures. Applications include simple retrieval, recommendation based on user profiles, exploration, and music analysis. Each application assumes specific usage scenarios in which the user plays the central role.

There are several categories into which applications can be classified. One is how much interaction they require. Roughly, there are two types. One type allows active exploration to discover new music or new insights into music. The other type is intended for more or less passive consumption of music. The first two applications in this chapter belong to the first category. The third application belongs to the second category.

Another category into which applications can be classified is the users' familiarity with the music. For example, does the application support the users in discovering something new? Or do they help the users manage their own (and thus to some extend familiar) collections? In this context an interesting question is also the size of the collections the applications are designed for. Some applications might work best for sizes of a few hundred pieces while others can (theoretically) deal with a few million.

Another difference is the level which the applications work in. In this chapter there are two levels, namely, the artist level and the track level. Some approaches can only work at the artist level (e.g. the web-based approach described in the previous chapter). Other approaches, such as the audio-based similarity can deal with any level (track, album, artist). In general, the type of similarity used (e.g. audio or web) also has a strong impact on the application and its usage scenarios. For example, audio-based similarity cannot be used to rate the quality of music,

Section	User Interaction	Size of the Collection	User's Familiarity with the Music	Type of Similarity	Level
3.2	active	small	any	audio	track
3.3	active	very large	unfamiliar	web	artist
3.4	passive	large	familiar	audio	track

Table 3.1: Rough categorization of applications in this chapter.

and thus requires additional precautions when using it to discover new music (to avoid flooding the user with low quality music).

This chapter primarily serves to describe some ideas on how similarity measures can be applied. The applications in this chapter can roughly be categorized as described in Table 3.1.

Related Work

There are a large number of applications besides those described in this chapter. For example, given a similarity measure for melody an application is query-by-humming and its variations (see e.g. [GLCS95]). An overview of MIR systems (many of which are applications of similarity measures) can be found in [TWV05]. Work specifically related to visualization of music collections is discussed in Subsection 3.2.2. Work related to representation of music collections at the artist level is discussed in Subsection 3.3.2 and work related to playlist generation in Subsection 3.4.2.

Structure of this Chapter

This chapter is structured as follows. In Section 3.2 an approach to organize and visualize music collections using a metaphor of geographic maps is described. The approach allows the user to interactively shift focus between different aspects of similarity. In Section 3.3 an approach to organize music collections at the artist level is described. In particular, artists are organized hierarchically into overlapping groups. These groups are automatically summarized with words co-occurring with the artists' names on web pages. In Section 3.4 an approach to dynamically generate playlists is described. Different heuristics are evaluated using hypothetical use cases. In Section 3.5 conclusions are drawn.

3.2 Islands of Music

The goal is to develop visualization tools which allow users to efficiently explore large music collections. Such visualizations would complement but not substitute current approaches which are more or less based on lists of items within a category

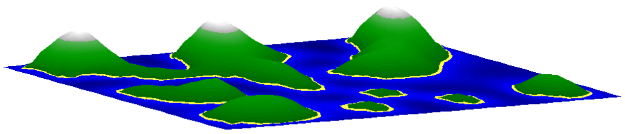


Figure 3.1: Islands of Music

(e.g. artists belonging to a genre). This section describes the Islands of Music visualization approach. In addition, an extension which enables the user to browse different views is described.

3.2.1 Introduction

The Islands of Music idea is to organize music collections on a map such that similar pieces are located close to each other. The structure is visualized using a metaphor of geographic maps. Figure 3.1 shows a 3-dimensional image of a map of 77 pieces of music. The idea is that each island represents a different style of music. Similar styles are connected by land passages. Mountains and hills on an island represent styles within styles, separated by valleys. These mountains and hills can be labeled with abstract terms describing low-level audio signal characteristics to help the user identify interesting regions. Furthermore, "weather charts" are used to describe the value of one attribute (e.g. bass energy) across the different regions of the map [Pam01; PRM02a].

This section describes the techniques used to compute Islands of Music. In addition, an extension is described which facilitates browsing different views [PGW03; PDW03a; Pam03; PDW04]. A view is defined by a specific aspect of similarity, e.g., rhythm or timbre. As the user gradually changes the focus between different views the map smoothly adapts itself by reorganizing the pieces and thus the structure of the islands. In particular, islands might merge, split, sink, or rise.

Limitations

Considering the original Islands of Music approach there are 3 major limitations.

A. Performance of the Similarity Measure
As discussed in the previous chapter, the performance of the audio-based similarity measures is far from perfect. In several cases the similarity matches are unacceptable. It is questionable if and how many such errors the user is willing to accept on a map where every piece is supposed to be close to all other pieces sounding similar. So far no user studies have been conducted to answer this question. This limitation is addressed in Section 3.4 where an application is suggested which does not rely on a perfect similarity matches.

B. Fuzziness of Music Similarity
One can argue that there is no single best place to map a piece of music on the map. In particular, a piece might exhibit different styles and might thus belong to several islands. Music similarity is inherently fuzzy. This limitation is addressed in Section 3.3 where the music collections are structured hierarchically (at the artist level) in such a way that an artist can be in different categories at the same time. This makes it easier to find artists which could belong to more than one category (or island).

C. Various Aspects of Similarity
As already mentioned in the previous chapter, there are various aspects of similarity. Thus it is of interest to allow the users to adjust the aspects they are interested in exploring. This is addressed in this section. However, this approach is still limited by the similarity measures which describe the aspects of similarity.

Structure of this Section

The remainder of this section is structured as follows. The next subsection gives an overview of related work regarding the organization and visualization of music collections. Subsection 3.2.3 describes the Self-Organizing Map (SOM) algorithm which is used to organize and map the music collections onto 2 dimensions. Subsection 3.2.4 describes visualizations which can be used in combination with SOMs. In Subsection 3.2.5 the Aligned-SOMs algorithm is described. This algorithm is used to align different views, each defined by a different aspect of similarity. In Subsection 3.2.6 a demonstration of browsing different views is given. Conclusions are drawn in Subsection 3.2.7.

3.2.2 Related Work

The metaphor of geographic maps can been used to visualize various types of information (including, e.g., visualization of large document archives [BWD02; Sku04]). In general the work presented in this section is related to many other approaches which visualize document collections and especially to those using Self-Organizing Maps (e.g. [KHLK98; MR00]).

An implementation of the Islands of Music which allows the creation of playlists by drawing lines on the maps was presented in [NDR05]. Hierarchical extensions were presented in [RPM02; Sch03].

Mörchen et al. [MUNS05] present an approach similar to Islands of Music. The authors use a variation of Self-Organizing Maps called Emergent Self-Organizing Maps. Instead of the Smoothed Data Histograms visualization they use the U-matrix to visualize clusters, they use a different color mapping where highly populated areas on the map are represented by valleys (instead of mountains), and they use toroidal maps to avoid border effects.

Lübbers [Lüb05] uses a metaphor of geographic maps in addition to an aural interface which allows the user to hear from a distance what music can be found in each direction. Similar work on aural interfaces for sounds was presented in [FB01].

Vignoli et al. [vGVvdW04] present an approach to visualize a music collection at the artist level. They use a variation of the Spring Embedder algorithm to optimize the layout of a graph. In [vGV05] the authors use this visualization to support playlist generation. A different approach using graphs to visualize music collections at the artist level was presented by Schedl et al. [SKW05a]. Further alternatives to visualize and explore music collections include the work presented in [CKGB02] which is based on the FastMap algorithm, the MusicSurfer application presented in [CKW+05], or the work presented in [THA04] which is based on metadata.

Related work which allows the user to adjust parameters of the similarity measure include [AP02a; BPS04; vGV05]. In [AP02a] the authors suggest a slider to control "Aha-effects". In particular, the slider can be used to control the interestingness of matches. An interesting match is, for example, if a piece sounds very similar but belongs to a different genre. In [BPS04] the users can control the weights on a similarity measure. The different similarity aspects that can be weighted are lyrics, sound, and style. In [vGV05] a system is presented where the users can move "magnets" to adjust the aspects of similarity they are interested in.

3.2.3 The Self-Organizing Map

The SOM [Koh82; Koh01] is an unsupervised neural network with applications in various domains including audio and music analysis (e.g. [CPL94; FG94; SP01; FR01]). As a clustering algorithm, the SOM is very similar to other partitioning algorithms such as k-means [Mac67]. In terms of topology preservation for visualization of high-dimensional data, alternatives include, e.g., Multi-Dimensional Scaling [KW78], Sammon's mapping [Sam69], and Generative Topographic Mapping [BSW98].

The SOM maps high-dimensional data to a 2-dimensional map in such a way that similar items are placed close to each other. The SOM consists of an ordered set of units which are arranged on a 2-dimensional grid (visualization space) called the map. Common choices to arrange the map units are rectangular or hexagonal grids. Each unit is assigned a model vector in the high-dimensional data space. A data item is mapped to the best-matching unit, which is the unit with the most similar model vector. The SOM can be initialized randomly, i.e., random vectors in the data space are assigned to each model vector. Alternatives include, for example, initializing the model vectors using the first two principal components of the data [Koh01].

After initialization, two steps are repeated iteratively until convergence (i.e., until there are no more significant changes in the mapping). The first step is to find the best-matching unit for each data item. In the second step the best-matching units and their neighbors are updated to better represent the mapped data. The neighborhood of each unit is defined through a neighborhood function and decreases with each iteration.

Details

To formalize the basic SOM algorithm, let $\mathbf{D}$ be the data matrix, let $\mathbf{M}_t$ be the model vector matrix, the $\mathbf{U}$ be the distance matrix, let $\mathbf{N}_t$ be the neighborhood matrix, let $\mathbf{P}_t$ be the partition matrix, and let $\mathbf{S}_t$ be the spread activation matrix.

The data matrix $\mathbf{D}$ is of size $n \times d$, where n is the number of data items, and d is the number of dimensions of the data. In case the data does not lie in a vector space the distance matrix can be used. This was done, for example, in [PDW03b]. For larger collections it is possible to reduce the dimensionality either by analyzing principal components or (in the case of very large collections) by random projections [Kas98; BM01].

The model vector matrix $\mathbf{M}_t$ is of size $m \times d$, where m is the number of map units. The values of $\mathbf{M}_t$ are updated in each iteration t. The matrix $\mathbf{U}$ of size $m \times m$ defines the distances between the units on the map. The neighborhood matrix $\mathbf{N}_t$ can be calculated, for example, as

$$\mathbf{N}_t = \exp\left(-\mathbf{U}^2/(2r_t^2)\right), \tag{3.1}$$

where r_t defines the neighborhood radius and monotonically decreases with each iteration. The matrix $\mathbf{N}_t$ is of size $m \times m$, symmetrical, with high values on the diagonal, and defines the influence of one unit on another. The sparse partition matrix $\mathbf{P}_t$ of size $n \times m$ is calculated given $\mathbf{D}$ and $\mathbf{M}_t$ as

$$\mathbf{P}_t(i,j) = \begin{cases} 1, & \text{if unit } j \text{ is the best match for item } i \\ 0, & \text{otherwise.} \end{cases} \tag{3.2}$$

The spread activation matrix $\mathbf{S}_t$, with size $n \times m$, defines the responsibility of each unit for each data item at iteration t and is calculated as

$$\mathbf{S}_t = \mathbf{P}_t \mathbf{N}_t. \tag{3.3}$$

At the end of each loop, the new model vectors $\mathbf{M}_{t+1}$ are calculated as

$$\mathbf{M}_{t+1} = \mathbf{S}_t^* \mathbf{D}_t \tag{3.4}$$

where $\mathbf{S}_t^*$ denotes the spread activation matrix, normalized so that the sum of each column equals one, except for units to which no items are mapped.

There are two main parameters for the SOM algorithm. One is the map size; the other is the final neighborhood radius. A larger map gives a higher resolution of the mapping but is computationally more expensive. The final neighborhood radius defines the smoothness of the mapping and should be adjusted depending on the noise level in the data.

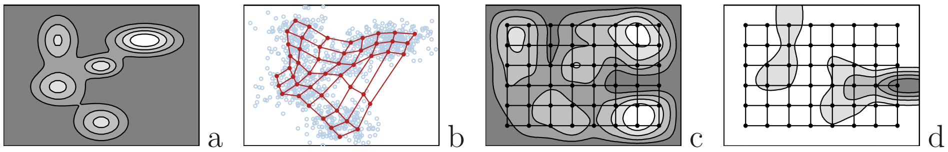


Figure 3.2: Illustration of the SOM. (a) The probability distribution from which the sample was drawn. (b) The model vectors of the SOM. (c) The SDH and (d) the U-matrix visualizations.

Illustrations

Figure 3.2 illustrates some important characteristics of the SOM. Samples are drawn from a 2-dimensional probability density function. A SOM (8×6) is trained so that the model vectors adapt to the topological structure of the data. There are two important characteristics of the non-linear adaptation. First, the number of data items mapped to each unit is not equal. Especially in sparse areas some units might represent no data items. Second, the model vectors are not equally spaced. In particular, in sparse areas the adjacent model vectors are relatively far apart while they are close together in areas with higher densities.

Both characteristics can be exploited to visualize the cluster structure of the SOM using smoothed data histograms (SDH) [PRM02b] and the U-matrix [US90], respectively. The SDH visualizes how many items are mapped to each unit. The smoothing is controlled by a parameter. The U-matrix visualizes the distance between the model vectors. The SDH visualization (Figure 3.2(c)) shows the cluster structure of the SOM. Each of the 5 clusters are identifiable. The U-matrix mainly reveals that there is a big difference between the clusters in the lower right and the upper right.

3.2.4 Visualizations

Various methods to visualize clusters based on the SOM have been developed. This subsection describes the Smoothed Data Histograms and other visualizations which are used for Islands of Music.

3.2.4.1 Smoothed Data Histograms

To create the Islands of Music metaphor the SDH visualization [PRM02b] can be used. The basic idea is that each data item votes for the map units which represent it best based on some function of the distance to the respective model vectors. All votes are accumulated for each map unit and the resulting distribution is visualized on the map.

A robust ranking function is used to gather the votes. The unit closest to a data item gets n points, the second n-1, the third n-2 and so forth, for the n closest map units. Basically the SDH approximates the probability density of the data on

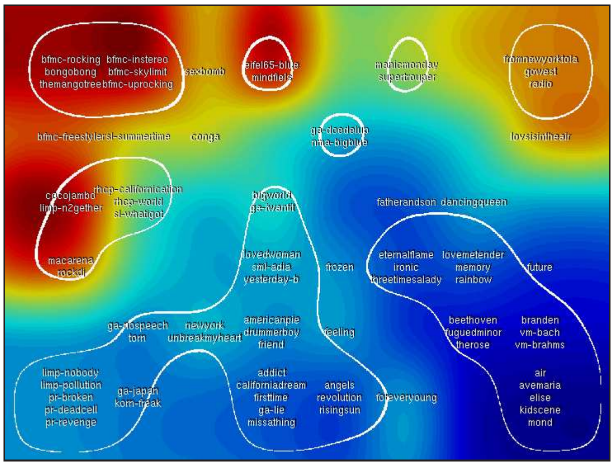


Figure 3.3: Weather chart metaphor for the distribution of bass energy on the map.

the map, which is then visualized using a specific color scale. A Matlab toolbox for the SDH is available online.[1] For details of the impact of the parameter n and a comparison to other cluster visualization techniques see [PRM02b]. Figure 3.2 demonstrates the SDH visualization using a toy example.

3.2.4.2 Weather Charts and Other Visualizations

Any attribute (such as an automatically extracted feature describing the bass energy of a piece, or genre metadata) can be used to describe different regions on the map. For example, landmarks such as mountains and hills can be labeled with descriptions which indicate what type of music can be found in the respective area. Details on the labeling of the Islands of Music can be found in [Pam01].

Alternatively a metaphor of weather charts can be used. For example, areas with a strong bass are visualized as areas with high temperatures, while areas with low bass correspond to cooler regions. Hence, for example, it becomes apparent that the pieces are organized so that those with a strong bass are in the west and those with less bass in the east. Figure 3.3 shows an example for a weather chart visualization using the same map shown in Figure 3.1. The contours of the islands are drawn to help the users keep their orientation.

Visualizing genre metadata can be very useful to evaluate the organization of a map. An example where a map computed for DB-ML (see Subsection 2.3.3 for a description of this collection) is visualized in terms of the genre distributions is shown in Figure 3.4. Noticeable are that some genres form very compact clusters such as punk. Others such as world are spread over the entire map.

[1] www.ofai.at/~elias.pampalk/sdh

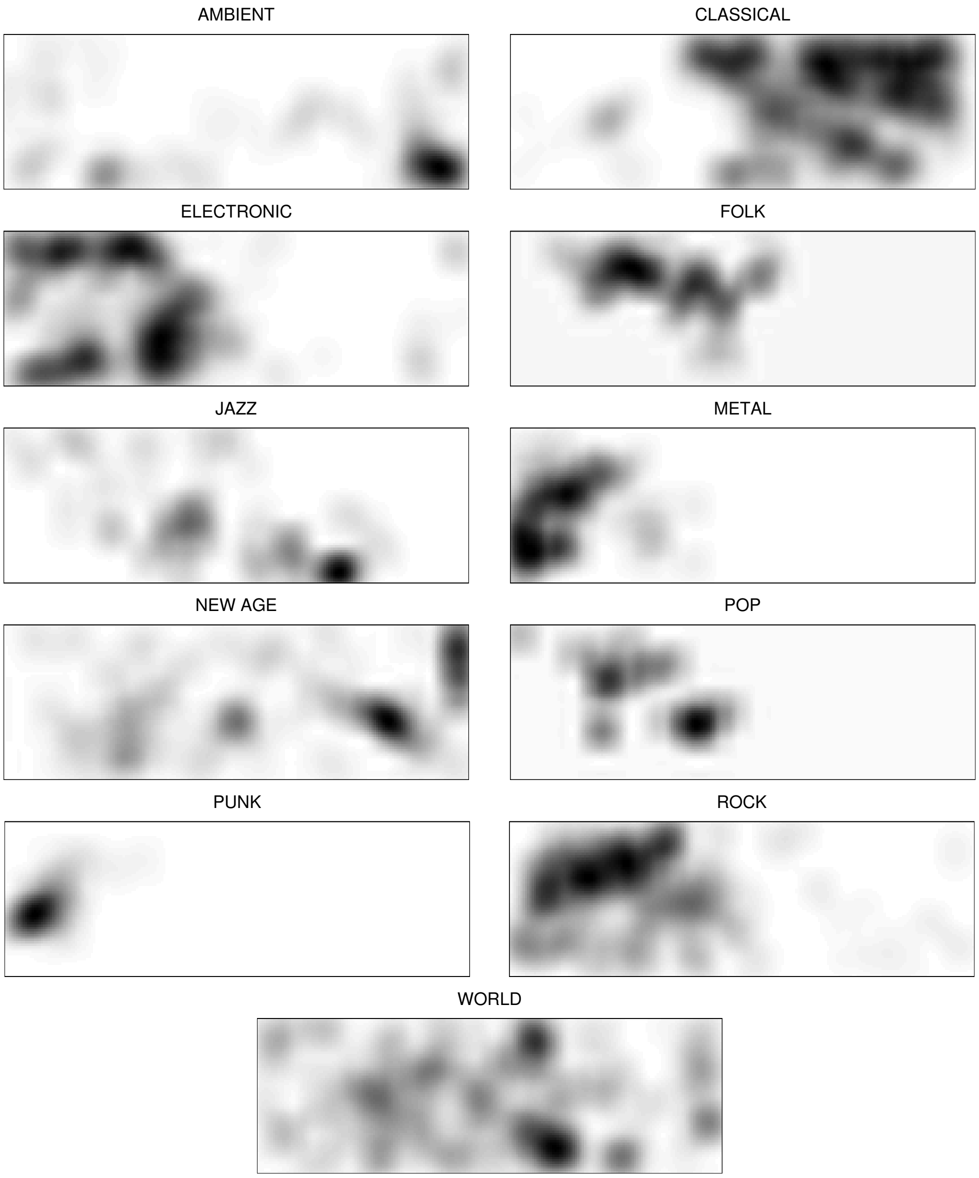


Figure 3.4: Distribution of genres on the SOM computed for DB-ML.

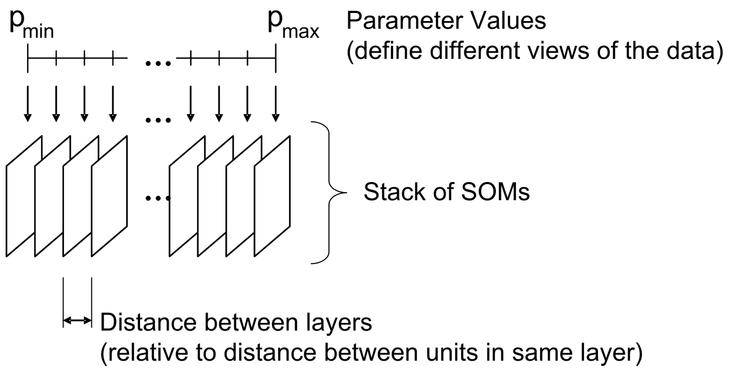


Figure 3.5: Aligned-SOM architecture.

3.2.5 Aligned Self-Organizing Maps

The SOM is a useful tool for exploring a data set given a similarity measure. However, when exploring music, the concept of similarity is not clearly defined, because there are many aspects to consider. Aligned-SOMs [PGW03; PDW03a; Pam03; PDW04] are an extension to the basic SOM that enable the user to interactively shift the focus between different aspects and explore the resulting gradual changes in the organization of the data.

The Aligned-SOMs architecture consists of several mutually constrained SOMs stacked on top of each other, as shown in Figure 3.5. Each map has the same number of units arranged in the same way (e.g. on a rectangular grid), and all maps represent the same pieces of music but organized with a different focus in terms of aspects of timbre or rhythm, for example.

The individual SOMs are trained such that each layer maps similar data items close to each other within the layer, and neighboring layers are further constrained to map the same items to similar locations. To that end, a distance between individual SOM layers is defined. This distance is made to depend on how similar the respective views are. The information between layers and different views of the same layer is shared based on the location of the pieces on the map. Thus, organizations from arbitrary sources can be aligned. Figure 3.6 shows the neighborhood radius which constraints units on the same map and units on neighboring maps. Each map has its own similarity space.

3.2.5.1 Details

The basic assumption is that the similarity measure is parametrized. For example, a parameter could control the weighting between rhythm and instrumentation related features. Given only one similarity parameter to explore $p \in \{p_{min} \cdots p_{max}\}$ a SOM is assigned to each of the extreme values. These two SOMs are laid on top of each other representing extreme views of the same data. Between these a

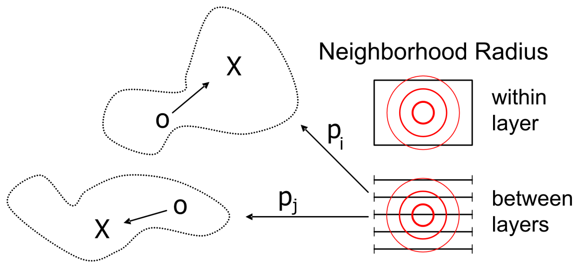


Figure 3.6: The Aligned-SOM neighborhood radius and updating a model vector (o) according to a data item (X) in different similarity spaces.

certain (manually defined, with computational restrictions) number of SOMs is inserted resulting in a stack of SOMs. Neighboring SOM layers in the stack represent slightly different parameter settings. All SOMs have the same map size and are initialized with the same orientation obtained, for example, by training the SOM in the middle of the stack and setting all other layers to have the same orientation. (This can be done by first using the assignment of pieces for the middle layer for all other layers. Second, all model vectors are initialized based on this assignment.)

To align the SOMs during training it is necessary to define a distance between layers which controls how smooth the transitions between layers are. Based on this distance the pairwise distances for all units in the stack are calculated and used to align the layers the same way the distances between units within a map are used to preserve the topology.

The (online) training is based on the normal SOM training algorithm. A data item and a layer are selected randomly. The best-matching unit for the item is calculated within the layer. The adaptation strength for all units in the stack depends on their distance to the best-matching unit. The model vectors within the selected layer are adapted as in the basic SOM algorithm. For adaptations in all other layers the respective representation of the data is used. The online training algorithm is outlined in Table 3.2. For the experiments described in this thesis a batch version of the Aligned-SOMs was used which is based on the batch training algorithm for individual SOMs and described in the next paragraphs.

The batch version of the Aligned-SOMs training algorithm is very similar to the batch SOM training algorithm described in Subsection 3.2.3. To train the SOM layers, the distance matrix $\mathbf{U}$ is extended to contain the distances between all units in all layers; thus the size of $\mathbf{U}$ is $ml \times ml$, where m is the number of units per layer, and l is the total number of layers. The neighborhood matrix is calculated according to Equation 3.1. For each aspect a of similarity, a sparse partition matrix $\mathbf{P}_{at}$ of size $n \times ml$ is needed. (In the demonstration discussed in Subsection 3.2.6, there are three different aspects: two are calculated from the spectrum and periodicity histograms, and one is based on meta-information.) The partition matrices for the first two aspects are calculated using Equation 3.2 with the extension that the best-matching unit for a data item is selected for each layer.

0	Initialize all SOM layers.
1	Randomly select a data item and layer.
2	Calculate best-matching unit c for the item within the layer.
3	For each unit (in all layers), a – Calculate adaptation strength using a neighborhood function given: – the distance between the unit and c, – the radius and learning rate for this iteration. b – Adapt the respective model vector using the vector space of the particular layer.

Table 3.2: Online training algorithm for Aligned-SOMs. Steps 1–3 are repeated iteratively until convergence.

Thus, the sum of each row equals the number of layers. The spread activation matrix $\mathbf{S}_{at}$ for each aspect a is calculated as in Equation 3.3. For each aspect a and layer i, mixing coefficients w_{ai} are defined with $\sum w_{ai} = 1$ that specify the relative strength of each aspect. The spread activation for each layer is calculated as

$$\mathbf{S}_{it} = \sum_{a} w_{ai}\mathbf{S}_{ait}. \tag{3.5}$$

Finally, for each layer i and aspect a with data $\mathbf{D}_a$, the updated model vectors $\mathbf{M}_{ait+1}$ are calculated as

$$\mathbf{M}_{ait+1} = \mathbf{S}^*_{it}\mathbf{D}_a, \tag{3.6}$$

where $\mathbf{S}^*_{it}$ is the normalized version of $\mathbf{S}_{it}$. In the demonstration in Subsection 3.2.6, the Aligned SOMs are initialized based on the meta-information organization for which only the partition matrix is given, which assigns each piece of music to a map unit. For the two views based on vector spaces, first the partition matrices are initialized, and then the model vectors are calculated from these.

Computation Time

The necessary resources in terms of CPU time and memory increase rapidly with the number of layers and depend on the degree of congruence (or incongruence) of the views. The overall computational load is of a higher order of magnitude than training a single SOM. For larger datasets, several optimizations are possible; in particular, applying an extended version of the fast winner search proposed by Kaski [Kas99] would improve the efficiency drastically, because there is a high redundancy in the multiple layer structure.

3.2.5.2 Illustration

Figure 3.7 illustrates the characteristics of Aligned-SOMs using a simple animal dataset [Koh01]. The dataset contains 16 animals with 13 boolean features describing appearance and activities such as the number of legs and the ability to swim. The necessary assumption for an Aligned-SOM application is that there is a parameter of interest which controls the similarity calculation. A suitable parameter for this illustration is the weighting between activity and appearance features.

To visualize the effects of this parameter 31 Aligned-SOMs are trained. The first layer uses a weighting ratio between appearance and activity features of 1:0. The 16th layer, i.e., the center layer, weights both equally. The last layer uses a weighting ratio of 0:1, thus, focuses only on activities. The weighting ratios of all other layers are linearly interpolated. The size for all maps is 3×4. The distance between layers was set to 1/10 of the distance between two adjacent units within a layer. Thus, the alignment between the first and the last layer is enforced with about the same strength as the topological constraint between the upper left and lower right unit of each map. All layers were initialized according to the organization obtained by training the center layer with the basic SOM algorithm.

From the resulting Aligned-SOMs 5 layers are depicted in Figure 3.7. The cluster structure is visualized using the SDH visualization. For interactive exploration a HTML version with all 31 layers is available online.[2] When the focus is only on appearance all small birds are located together in the lower right corner of the map. The Eagle is an outlier because it is bigger than the other birds. All mammals are located in the upper half of the map separating medium sized ones on the left from large ones on the right.

As the focus is gradually shifted to activity descriptors the organization changes smoothly. When the focus is solely on activity the predators are located on the left and others on the right. Although there are several significant changes regarding individuals, the overall structure has remained the same, enabling the user to easily identify similarities and differences between two different ways of viewing the same data.

3.2.6 Browsing Different Views (Demonstration)

To demonstrate the Aligned-SOMs applied to music collections a HTML-based interface was implemented [PDW03a]. Three different aspects are combined. One is based on periodicity histograms which can be related to rhythmic characteristics. The second aspect is based on spectrum histograms which relate to timbre. (Periodicity histograms and spectrum histograms are described in [PDW03a].) The third aspect is based on metadata. In this specific case the pieces were manually placed on the map. This aspect is referred to as the "user defined aspect" below. Any other placement could be used (e.g. using album release date on the x-axis and beats per minute on the y-axis).

A screenshot is shown in Figure 3.8. An online demonstration is available.[2] For this demonstration a small collection of 77 pieces from different genres was

[2]http://www.ofai.at/~elias.pampalk/aligned-soms

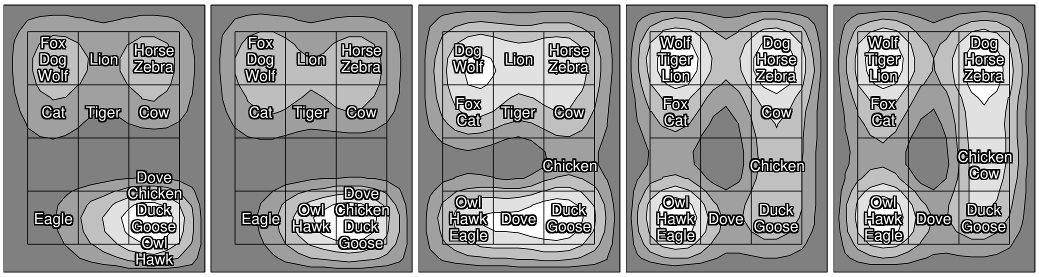


Figure 3.7: Aligned-SOMs trained with the animal dataset. From left to right the figures are (a) the first layer with weighting ratio 1:0 between appearance and activity features, (b) 3:1, (c) 1:1, (d) 1:3, (e) 0:1. The shadings represent the density calculated using SDH ($n = 2$ with bicubic interpolation). White corresponds to high densities, gray to low densities.

used. Although real world music collections are significantly larger, in some cases even small numbers can be of interest as they might occur, for example, in a result set of a query for the top 100 in the charts. The limitation in size is mainly due to the simple HTML interface. Larger collections would require an hierarchical extension to the interface. For example, one approach would be to represent each island only by the most typical member, and allow the user to zoom in and out of the map.

The user interface (Figure 3.8) is divided into 4 parts: the navigation unit, the map, and two codebook visualizations. The navigation unit has the shape of a triangle, where each corner represents an organization according to a particular aspect. The user defined view is located at the top, periodicity on the left, and spectrum on the right. The user can navigate between these views by moving the mouse over the intermediate nodes. In total there are 73 different nodes the user can browse.

The current position in the triangle is marked red (the red marker is set to the top corner in the screenshot). Thus, the current map displays the user defined organization. For example, all classical pieces in the collection are grouped together in the upper left. On the other hand, the island in the upper right of the map represents pieces by Bomfunk MCs. The island in the lower right contains a mixture of different pieces by Papa Roach, Limp Bizkit, Guano Apes, and others which are partly very aggressive. The other islands contain more or less arbitrary mixture of pieces, although the one located closer to the Bomfunk MCs island contains music with stronger beats.

Below the map are the two codebook visualizations, i.e., the model vectors for each unit. This allows the user to interpret the map and primarily serves research purposes. In particular, the codebooks explain why certain pieces are located in a specific region and what the differences between regions are. The codebook visualizations reveal that the user defined organization is not completely

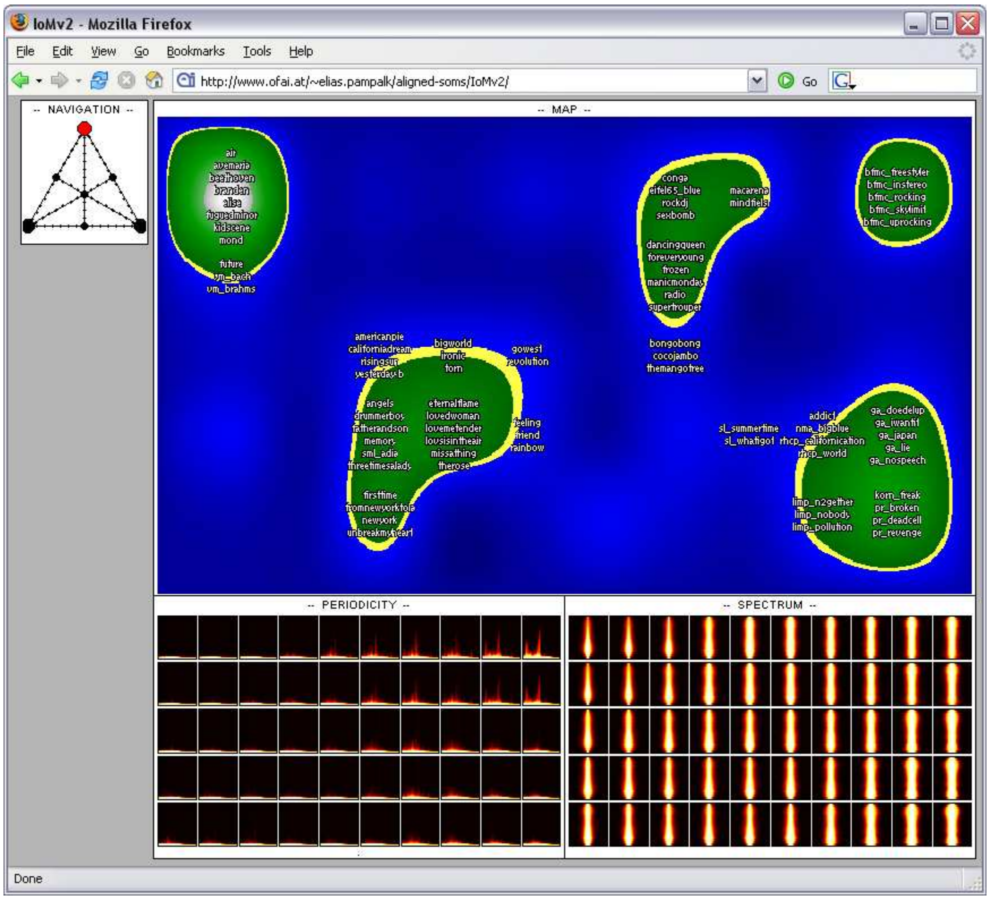


Figure 3.8: Screenshot of the HTML-based user interface. The navigation unit is located in the upper left, the map to its right, and beneath the map are the codebook visualizations where each subplot represents a unit of the 10×5 SOM trained on 77 pieces of music. On the left are the periodicity histogram codebooks. The x-axis of each subplot represents the range from 40 (left) to 240bpm (right) with a resolution of 5bpm. The y-axis represents the strength level of a periodic beat at the respective frequency. The color shadings correspond to the number of frames within a piece that reach or exceed the respective strength level at the specific periodicity. On the right are the spectrum histogram codebooks. Each subplot represents a spectrum histogram mirrored on the y-axis. The y-axis represents 20 frequency bands while the x-axis represents the loudness. The color shadings correspond to the number of frames within a piece that reach or exceed the respective loudness in the specific frequency band.

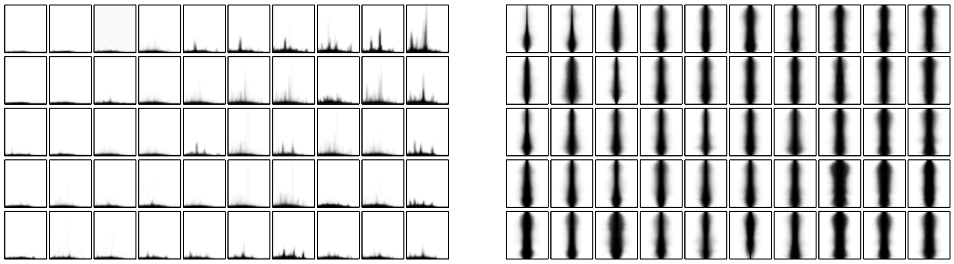


Figure 3.9: Codebooks of the organization focusing solely on the periodicity histograms. On the left is the codebook of the periodicity histograms, on the right the codebook of the spectrum histograms. See Figure 3.8 for a description of the axes.

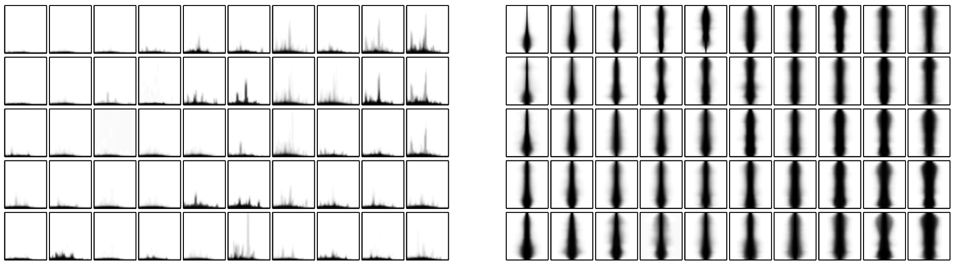


Figure 3.10: Codebooks of the organization focusing solely on the spectrum histograms. See Figure 3.8 for a description of the axes.

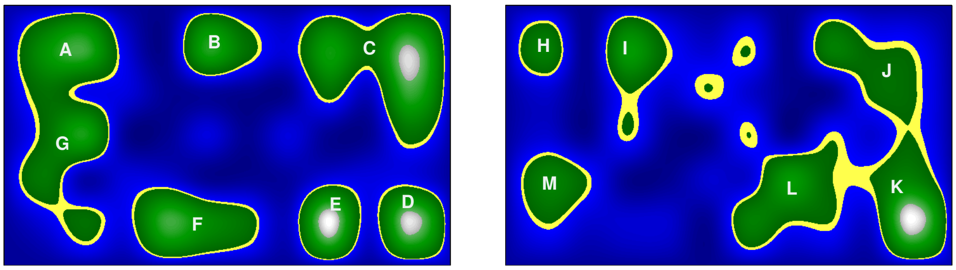


Figure 3.11: Two extreme views of the data and the resulting Islands of Music. On the left the focus is solely on the periodicity histograms, on the right the focus is solely on spectrum histograms.

arbitrary with respect to the features extracted from the audio. For example, the periodicity histogram has the highest peaks around the Bomfunk MCs island and the spectrum histogram has a characteristic shape around the classical music island. This characteristic shape occurs when most of the energy is in the low and mid frequencies. The shadings are a result of the high variations in the loudness, while the overall relatively thin shape is due to the fact that the maximum level of loudness is not constantly reached.

The codebooks of the extreme perspectives are shown in Figures 3.9 and 3.10. When the focus is solely on periodicity the model vectors of the SOM can better adapt to variations between histograms and thus represent the histograms with higher detail. The same applies for the codebook representing the spectrum histograms. Also noticeable is how the organization of the model vectors changes as the focus is shifted. While there is no obvious global structure in the organization of the spectrum histograms when the focus is on periodicity, the organization becomes apparent if the focus is changed to spectrum information.

An important characteristic of Aligned-SOMs is the global alignment of different views. In this demonstration the user defined organization forces the periodicity patterns of music by Bomfunk MCs to be located in the upper right. If trained individually, these periodicity histograms would be found on the lower right which is the furthest region from the upper left where pieces such as, e.g., Für Elise by Beethoven can be found.

Figure 3.11 shows the shapes of the islands for the two extreme views focusing only on spectrum or periodicity. When the focus is on spectral features the island of classical music (upper left) is split into two islands where “H” represents piano pieces and “I” orchestra. Inspecting the codebook reveals that the difference is that orchestra music uses a broader frequency range. On the other hand, when the focus is on periodicity a large island is formed which accommodates all classical pieces on one island “A”. This island is connected to island “G” where also non-classical music can be found such as the song Little Drummer Boy by Crosby & Bowie or Yesterday by the Beatles. Although there are several differences between the maps the general orientation remains the same. For example, the island in the upper right always accommodates the pieces by Bomfunk MCs.

3.2.7 Conclusions

This section described the Islands of Music idea and an extension which allows the users to adjust the aspect of similarity they are interested in. This allows interactive exploration of music collections by browsing different views. The Aligned-SOM algorithm performs well on the toy example used and also on a different application where the task is to analyze expressive piano performances [PGW03; GPW04]. However, the results for the music collections are not very convincing. The main reason for this is that the performance of the similarity measures used to describe different aspects of music is very limited.

Nevertheless, Aligned-SOMs can be used to compare two similarity measures (e.g. [PDW03b]). Furthermore, they are not limited to the use of audio-based similarity measures. Any form of similarity could be used (e.g. similarity defined by experts). In addition, future improvements of audio-based similarity will make the mapping of pieces into 2 dimensions more reliable. Perhaps it will be possible to reach a level which will make it interesting to conduct user studies.

Alternatively, one approach is to avoid giving the user the impression that the few pieces located close to each other are all of the similar pieces in the collection. This will be dealt with in the next section. A second alternative is to develop applications where, from a user's perspective, some variance is actually desirable. This will be dealt with in the Section 3.4.

3.3 Fuzzy Hierarchical Organization

This section describes an approach to hierarchically organize music collections at the artist level. Artists are organized into overlapping groups according to similarity. The reason for dealing with artists instead of tracks is that the similarities are computed by analyzing the content of web pages ranked by Google (as described in Section 2.5) which can only be done at the artist level. However, any similarity measure can be used, also one which allows computation of the similarities at the track level.

Furthermore, in this section the advantages and disadvantages of different strategies to automatically describe these clusters of artists with words are analyzed. A domain specific dictionary is used to improve the results. An HTML-based demonstration is available.[3] This section describes the work presented in [PFW05a].

[3]http://www.ofai.at/~elias.pampalk/wa

3.3.1 Introduction

Hierarchical structures are necessary when dealing with large collections. The approach described in this section uses a fuzzy hierarchical organization. This allows an artist to be located in several branches of the hierarchy.

As mentioned in Subsection 3.2.1 music is inherently fuzzy. A piece of music can exhibit different styles making it impossible to pigeonhole it. One approach to deal with this is to use overlapping clusters. For example, if the first level in the hierarchy gives the user the choice between rock and electronic music, it would be difficult to find rock pieces with electronic influences (or the other way round). In particular, it would require the users to search all possible branches to ensure that they have not missed artists of interest (which is not the purpose of a hierarchical organization).

In addition to the fuzziness of music, the performance of the similarity measure is suboptimal. Rock pieces might wrongly be classified into the electronic branch. However, for such pieces the second choice might have been to place them in the rock branch. Introducing fuzziness (thus increasing the number of pieces per branch) allows the system to correct false negatives. Furthermore, this avoids giving the user a wrong impression about the accuracy of the structure.

A very important issue for hierarchical interfaces is how to summarize the contents of a branch. That is, how can a group of similar artists be summarized and described to the user. A good summary should be very short. A user should be able to quickly read (or listen to) several summaries. Subsection 3.3.4 describes an approach which uses a domain specific dictionary (i.e. a dictionary with terms used to described music) to create summaries based on words which occur on web pages containing the artists' names.

Structure of this Section

This section is structured as follows. The next subsection gives an overview of the background. In particular, some currently existing systems are briefly described. Subsection 3.3.3 describes the fuzzy hierarchical clustering algorithm used to structure the data. Subsection 3.3.4 describes the techniques used to select words to describe the clusters of artists. Subsection 3.3.5 describes and discusses the results obtained for a music collection with 224 artists. Finally, conclusions are drawn in Subsection 3.3.6.

3.3.2 Background

Approaches to organize music collections at the artist level exist and are in use. One common approach is to use genres. Although such organizations are very common they have limitations. As already discussed in Subsection 2.3.2 genre taxonomies are inconsistent. There are usually no clear boundaries between genres and many artists create music which fits to more than just one genre. In some cases

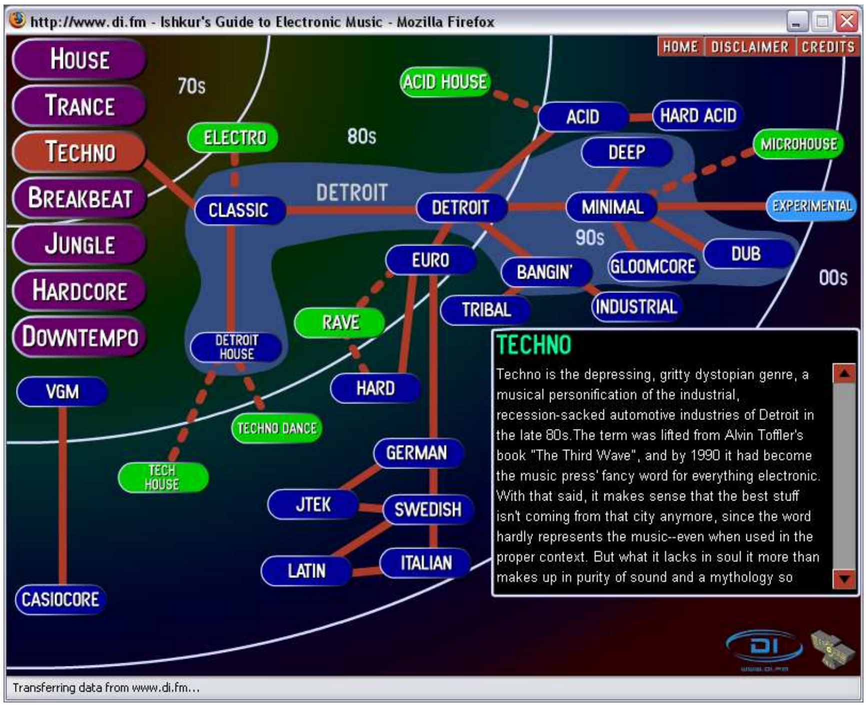


Figure 3.12: Screenshot of Ishkur's Guide to Electronic Music showing subgenres for techno music.

an expert is required to correctly classify an artist into one of hundreds of genres and subgenres. For example, Ishkur's Guide to Electronic Music distinguishes 180 different genres just for electronic music.[4] A screenshot of this guide is shown in Figure 3.12. In such cases it is questionable how useful the genre labels (without further information) are for non-experts.

The main advantage of organizing music into genres is that it enables users to easily find similar music. In particular, if the users know that their favorite artist belongs to a certain genre, they might want to explore other artists belonging to the same genre. However, finding similar artists does not require categorization into genres. For example, Amazon[5] (Figure 3.13) and the All Music Guide[6] (Figure 3.14) offer lists of similar artists to explore.

Amazon customers can browse genres (or categories) if they like. However, once they searched (or selected) a specific artist this option is not displayed on the artist's web page. Similarly, All Music Guide categorizes artists into genres and styles. However, the list of similar artists is placed easily accessible next to the list of genres and styles on the web page.

[4]http://www.di.fm/edmguide
[5]http://amazon.com
[6]http://allmusic.com

Figure 3.13: Screenshot of the information Amazon displays for an artist (Ben Harper in this case). In the lower left of the screen is a list of similar artists.

Figure 3.14: Screenshot of the information the All Music Guide displays for an artist (Ben Harper in this case). In the lower center of the screen is a list of similar artists.

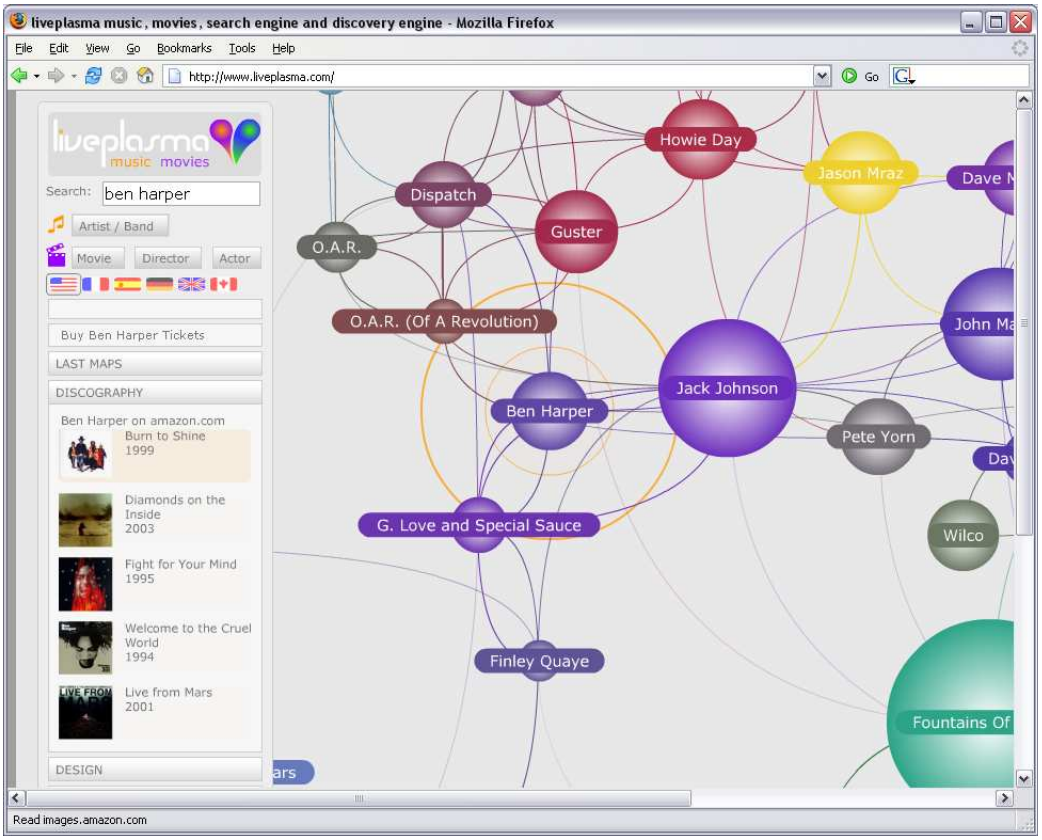


Figure 3.15: Screenshot of Liveplasma with the query results for “Ben Harper”.

Another available system which visually organizes music at the artist level is Liveplasma.[7] A screenshot is shown in Figure 3.15. Each artist is represented by a disc. The size of the disc corresponds to the popularity. Lines connect similar artists.

In contrast to these commercial systems this section describes an approach which does not display artist names to the user. The assumption is that the user does not know the artist names. Furthermore, the users do not need to start with a familiar artist. Another difference is that the approach described in this section does not require knowledge of customers' shopping behavior or experts to annotate the data. Instead the similarities are computed using common web pages.

The same hierarchical approach used here to organize artists has been previously applied to organize large collections of drum sample libraries [PHH04]. The basic idea is similar to the approach used by the search engine vivisimo.[8] Other approaches which organize music collections at the artist level include [vGVvdW04] and [SKW05a].

3.3.3 Hierarchical Clustering

The user interface defines the requirements for the clustering algorithm. The user interface is shown in Figure 3.19. Instead of complex graphical visualizations simple lists of words are used to present the information. The user interface supplies room for 5 nodes per level of the hierarchy. This number is more or less arbitrary and can easily be changed. However, the important point is that the number is fixed and does not vary.

A clustering technique which fulfills these simple requirements is the one-dimensional Self Organizing Map which can be structured hierarchically [Mii90; KO90]. More flexible approaches include the Growing Hierarchical SOM [DMR00] and variations (e.g. [PWC04]). Other alternatives include, for example, Hierarchical Agglomerative Clustering as used in [PWL01].

The SOM groups similar items into clusters and places similar clusters close to each other. To create an overlap between clusters the size of each cluster is increased by 20% after training the SOM. This is done by adding artists closest to the model vector. This overlap makes it easier to find artists which are on the border of two or more clusters.

Recursively, for each cluster another one-dimensional SOM is trained (for all artists assigned to the cluster) until the cluster size falls below a certain limit (e.g. if less than 7 artists remain). Figure 3.16 shows the basic architecture. In [PHH04] the same one-dimensional overlapping clustering approach and a similar user interface was used to organize large drum sample libraries. The feedback gathered from the users was very positive.

[7] http://liveplasma.com
[8] http://www.vivisimo.com

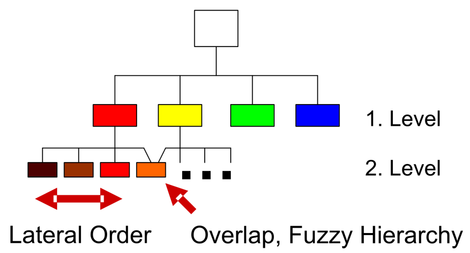


Figure 3.16: Basic architecture of the fuzzy hierarchy.

Evaluation of the Clustering

The easiest way to evaluate the cluster hierarchy is to test whether artists from the same genre are grouped together. For the evaluation described here a collection of 224 artists was used. A list of the artists is online.[9] Each of the 14 genres is represented by 16 artists. The genres are listed in Figure 3.17.

Figure 3.17 shows the distribution of the genres within the nodes (i.e. clusters) in the hierarchical structure. (A screenshot of how the hierarchy is displayed in the user interface is shown in Figure 3.19.) At the first level classical music (node n1) is well separated from all other music. The effects of the overlap are immediately visible as the sum of artists mapped to all units in the first layer is beyond 224. One example for a direct effect of the overlap is that there is jazz music in node n1, which would not be there otherwise. The nodes n1 and n5 are the only ones at the first level containing jazz music. Electronic and rap/hip-hop is only contained in n2 and n3, and blues only in n4 and n5.

At the second level most nodes have specialized. For example, n5.1 contains 34 artists mainly from jazz and blues and few from rock & roll. Another nice example is n3.2 which contains mostly punk but also some alternative. An interesting observation is that the one-dimensional ordering of the SOM (i.e. similar units should be close to each other) is not apparent. One reason for this might be the extremely high-dimensional data (as mentioned in Subsection 2.5.2, 200 eigenvectors are necessary to preserve 95% of the variance in the data). Furthermore, noticeable is that there are some nodes which seem to contain a bit of almost every genre.

Figure 3.18 shows what happens at the third level (in the subbranch of node n2). For example, while node n2.3 contains artists from punk and soul/R&B none of its children mix the two. Another positive example is that node n2.5.1 captures most of the electronic music in n2.5. However, as can be seen the clustering is far from perfect and leaves a lot of room for improvement. These deficiencies can be traced back to the performance of the similarity measure.

[9] http://www.cp.jku.at/people/knees/artistlist224.html

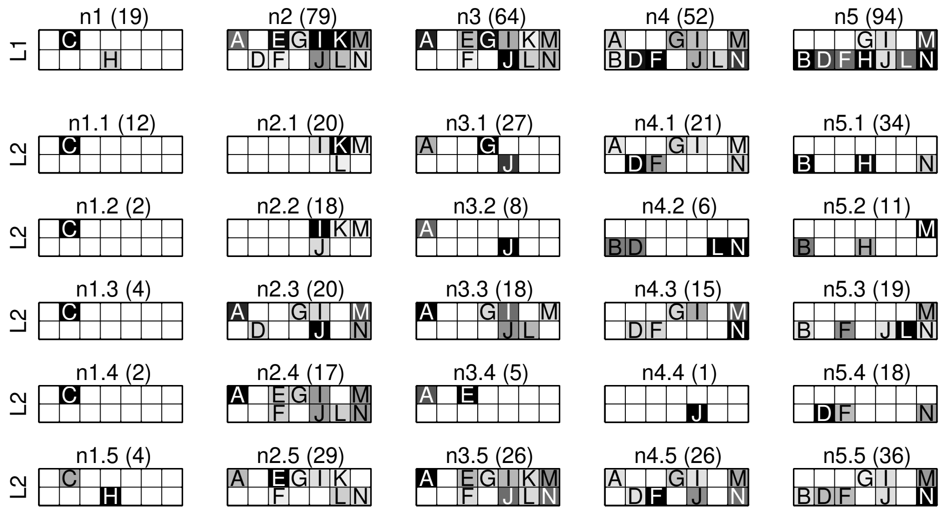


Figure 3.17: Distribution of the 14 genres in the nodes of the first level (L1, first row) and second level (L2, all columns starting with the second row). For example, n2.4 (L2 node) is the fourth child of parent n2 (L1 node). Each subplot represents the distribution of genres in a node (visualized as histogram and displayed in two lines to save space). Black corresponds to high values. The boxes in the histogram correspond to the following genres: A alternative/indie, B blues, C classic, D country, E electronic, F folk, G heavy, H jazz, I pop, J punk, K rap/hip-hop, L reggae, M R&B/soul, N rock & roll. The numbers in brackets are the number of artists mapped to the node.

3.3.4 Term Selection for Cluster Description

Term selection is a core component of the user interface. The goal is to select words which best summarize a group of artists. The approach described here is based on three assumptions with respect to the user interaction. First, the artists are mostly unknown to the user (otherwise it would be possible to label the nodes with the artists' names). Second, it is unknown which artists the user knows (otherwise these could be used to describe the nodes). Third, the assumption is made that space is very limited and thus each node should be described with as few words as possible. Dropping the second assumption could lead to very interesting interactive user interfaces. However, this is beyond the scope of this thesis.

In the experiments described here five term selection techniques are compared. In addition, two different approaches with respect to the set of terms to start with in the first place are compared. In addition to the straightforward approach to use the terms which are also used for the similarity computations, an approach based on a domain-specific dictionary is used.

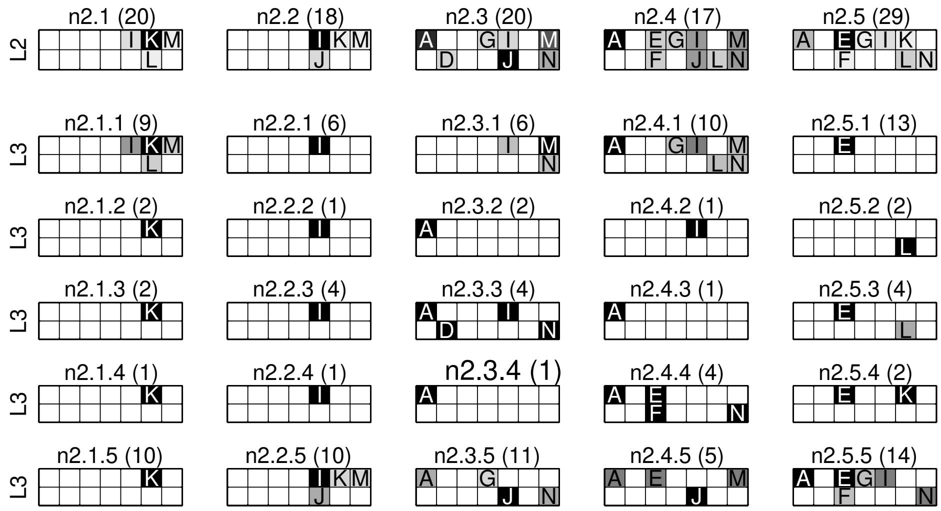


Figure 3.18: Distribution of the 14 genres in the nodes of the second (L2) and third (L3) level in the subbranch of node n2.

3.3.4.1 Techniques

Given are the term frequency tf_{ta}, document frequency df_{ta}, and the Cosine normalized $tf \times idf$ weight w_{ta} for each term t and artist a (see Subsection 2.5.2). A straightforward approach is to use the $tf \times idf$ computations, i.e. w_{ta}. For each node (i.e. cluster c) the average of the assigned artists is computed $w_{tc} = 1/|c| \sum_{a \in c} w_{ta}$, and the terms with the highest values are selected.

The second approach is called "LabelSOM" [Rau99] and has successfully been applied to label large document collections organized by SOMs. LabelSOM is built on the observation that terms with a very high w_{tc} and a high variance (i.e., they are very rare in some of the documents in the cluster) are usually poor descriptors. Thus, instead of w_{tc} the variance of w_{ta} in c is used to rank the terms (better descriptors have lower variances). Since terms which do not occur in c ($w_{tc} = 0$) have variance 0, terms with w_{tc} below a manually defined threshold are removed from the list of possible candidates. This threshold depends on the number of input dimensions and how the vectors are normalized. In the experiments described here the input dimension is 4139 (see Subsection 2.5.2) and a threshold of 0.045 is used. For the approach with the dictionary (see below) where the input dimension is 1269 a threshold of 0.1 is used. In cases where a cluster consists of only on artist the $tf \times idf$ ranking is used instead.

Neither $tf \times idf$ ranking nor LabelSOM try to find terms which discriminate two nodes. However, emphasizing differences between nodes of the same parent helps reduce redundancies in the descriptions. Furthermore, the assumption can be made that the user already knows what the children have in common after reading the description of the parent. A standard technique to select discriminative terms is the χ^2 (chi-square) test (e.g. [YP97]). The χ^2-value measures the independence of t from group c and is computed as,

$$\chi^2_{tc} = \frac{N(AD - BC)^2}{(A+B)(A+C)(B+D)(C+D)} \tag{3.7}$$

where A is the number of documents in c containing t, B the number of documents not in c containing t, C the number of documents in c without t, D the number of documents not in c without t, and N is the total number of retrieved documents. As N is equal for all terms, it can be ignored. The terms with highest χ^2_{tc} values are selected because they are least independent from c. Note that the document frequency is very informative because df_{ta} describes the percentage of times the terms occur in the 50 retrieved documents per artist (as described in Subsection 2.5.2).

The fourth technique was proposed by Lagus and Kaski (LK) [LK99]. Like LabelSOM it was developed to label large document collections organized by SOMs. While χ^2 only uses df, LK only use tf. The heuristically motivated ranking formula (higher values are better) is,

$$f_{tc} = (tf_{tc} / \sum_{t'} tf_{t'c}) \cdot \frac{(tf_{tc} / \sum_{t'} tf_{t'c})}{\sum_{c'} (tf_{tc'} / \sum_{t'} tf_{t'c'})}, \tag{3.8}$$

where tf_{tc} is the average term frequency in cluster c. The left side of the product is the importance of t in c defined through the frequency of t relative to the frequency of other terms in c. The right side is the importance of t in c relative to the importance of t in all other clusters.

The fifth approach is a variation of LK. It serves to demonstrate the effects of extreme discrimination. In particular, in this variation tf_{tc} are normalized over the whole collection such that a word which occurs 100 times in cluster c and never in any cluster is equally important to a word that occurs once in c and never otherwise. As shown later this approach can only produce meaningful results when used in combination with a specialized dictionary. All terms which do not occur in at least 10% of the documents per cluster are ignored. The ranking function (higher values are better) is,

$$f_{tc} = (tf_{tc} / \sum_{c'} tf_{tc}) \cdot \frac{(tf_{tc} / \sum_{c'} tf_{tc'})}{\sum_{c''} (tf_{tc''} / \sum_{c'} tf_{tc'})}. \tag{3.9}$$

In addition two combinations are implemented. In particular combining LK with χ^2, and the LK variant with χ^2. In both cases the values are combined by multiplication.

3.3.4.2 Domain-Specific Dictionary

One of the main pillars of this section is the use of a dictionary to avoid describing clusters with artist, album, and other specialized words likely to be unknown to the user. This dictionary contains general words used to describe music, such as genre names. The dictionary contains 1398 entries,[10] 1269 of these occur in the retrieved documents. The dictionary was manually compiled by the author in a sloppy manner by copying lists from various sources such as Wikipedia, the Yahoo directory, allmusic.com, and other sources which contained music genres (and subgenres), instruments, or adjectives. The dictionary is far from complete and contains terms which should be removed (e.g. *world*, *uk*, *band*, and *song*). However, to get a better understanding of possible drawbacks the original version of the dictionary was not modify.

Each retrieved web page is parsed and the term frequencies, document frequencies, and *tf* $\times$ *idf* are computed. A question that arises is why the dictionary was not used in the first place to compute the similarities. There are two reasons.

First, the classification performance using k-nearest neighbors with leave-one-out validation is only about 79% compared to the 85% of the standard approach. Considering the size of the collection this might not be significant. However, the explanation for this is that the standard approach captures a lot of the very specific words such as the artists' names, names of their albums and many other terms which co-occur on related artist pages.

Second, while the dictionary is an important pillar of this approach an effort is made not to rely too much upon it. By manipulating the dictionary it is very likely that 100% classifcation accuracies can be achieved on the set of 224 artists. However, such results could not be generalized to other music collections. Furthermore, in the current approach the specialized dictionary can be replaced at any time without impact on the hierarchical structure.

Limitations

As already mentioned the dictionary has some limitations. For example, it includes terms that do not help summarize clusters such as *song*, *uk*, *band*, *world*, and *musical*. As can be seen in Tables 3.3 and 3.4 these words occur very frequently. On the other hand, some terms were completely forgotten such as *love* and *hate*. Another problem are equivalences which are not dealt with. Examples include: *hip-hop* and *hip hop*, *rock and roll* and *rock & roll*, *rock and pop* and *pop/rock*. Furthermore, no attempt is made to deal with different languages, the dictionary contains only English terms.

3.3.5 Results and Discussion

In this subsection first the implemented user interface is described. Second, different term selection techniques are compared. This is done using simple lists of words (see Tables 3.3 and 3.4). Finally, the approach is discussed.

[10] http://www.ofai.at/~elias.pampalk/wa/dict.txt

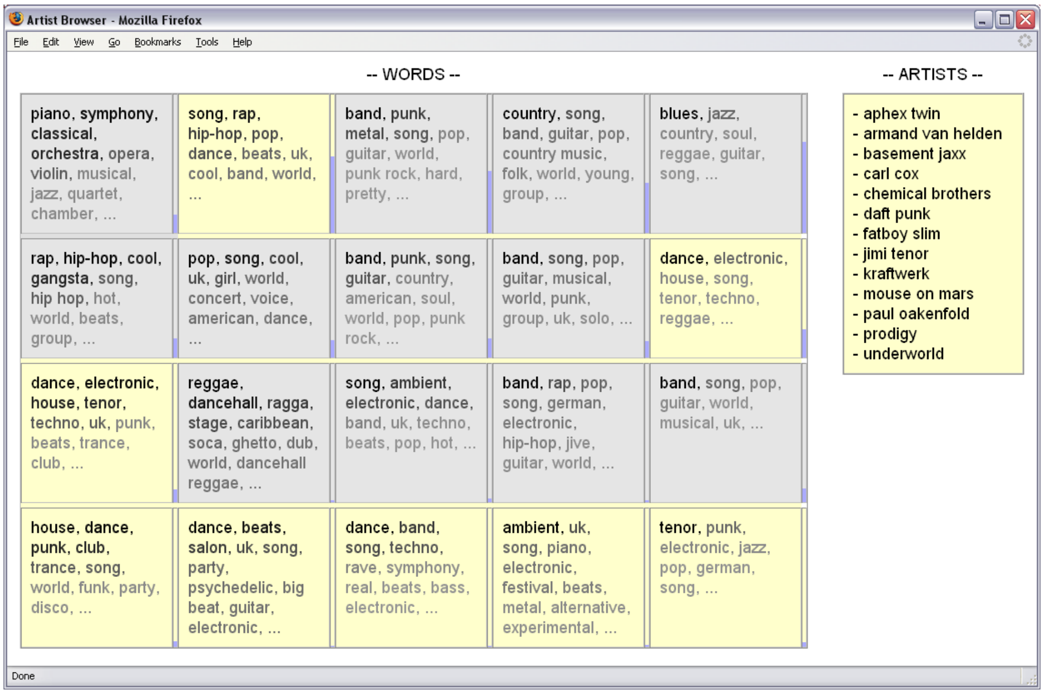


Figure 3.19: Screenshot of the HTML user interface.

3.3.5.1 User Interface

To demonstrate the approach a very simple HTML interface was implemented.[11] There are two parts of the interface: the hierarchy of clusters visualized as a grid of boxed texts and, just to the right of it, a display of a list of artists mapped to the currently selected cluster. The clusters of the first level in the hierarchy are visualized using the five boxes in the first (top) row. After the user selects a cluster, a second row appears which displays the children of the selected cluster. The selected clusters are highlighted in a different color. The hierarchy is displayed in such a way that the user can always see every previously made decision on a higher level. The number of artists mapped to a cluster is visualized by a bar next to the cluster. Inside a text box, the highest ranked terms are displayed. In general the top 10 are shown, however, if a term's value is below 10% of the highest value then it is not displayed.

The value of the ranking function for each term is coded through the color in which the term is displayed. The best term is always black and as the values decrease the color fades out. For debugging purposes it is also possible to display the list of all ranked words for a cluster. Figure 3 shows what the user interface looks like (using LK labeling) after node n2.5.1 was selected (thus 4 levels are visible).

[11] http://www.ofai.at/~elias.pampalk/wa

	with dictionary	without dictionary
	tf × *idf*	
n1	classical, piano, orchestra, symphony, musical	classical, piano, orchestra, works, composer
n2	song, pop, world, uk, band	listen, color, news, pop, size
n3	song, band, pop, world, guitar	band, listen, great, pop, live
n4	song, band, guitar, pop, world	color, listen, live, pop, size
n5	song, blues, band, guitar, world	color, size, family, listen, blues
	LabelSOM	
n1	world, musical, concert, song, uk	two, information, musical, recordings, great
n2	world, musical, classical, song, real	content, know, people, listen, sound
n3	musical, world, song, group, pop	great, sound, news, listen, live
n4	musical, world, classical, song, pop	content, great, information, pop, listen
n5	musical, classical, group, song, world	information, great, content, listen, pop
	χ^2	
n1	piano, orchestra, symphony, concert, opera	classical, composer, musical, great, piano
n2	dance, rap, hip-hop, beats, group	news, pop, sound, track, release
n3	guitar, musical, group, punk, metal	band, sound, live, great, pop
n4	musical, guitar, country, group, blues	live, band, pop, news, policy
n5	blues, band, country, pop, jazz	blues, jazz, country, hits, policy
	Lagus & Kaski	
n1	piano, symphony, classical, orchestra, opera	op, bach, piano, symphony, classical
n2	song, rap, hip-hop, pop, dance	hop, hip, rap, listen, pop
n3	band, punk, metal, song, pop	band, punk, metal, bands, great
n4	country, song, band, guitar, pop	country, alice, elvis, brooks, rate
n5	blues, jazz, country, soul, reggae	blues, jazz, color, john, size
	$\chi^2\cdot$ LK	
n1	piano, orchestra, symphony, opera, violin	classical, piano, composer, orchestra, symphony
n2	rap, dance, hip-hop, beats, uk	news, pop, hop, hip, track
n3	punk, guitar, metal, musical, group	band, punk, live, sound, great
n4	country, guitar, musical, group, blues	country, live, band, pop, hits
n5	blues, jazz, country, band, soul	blues, jazz, country, john, hits
	Lagus & Kaski variant	
n1	rondo, fortepiano, contralto, fugue, mezzo	nabucco, leopold, cycles, figaro, sonatas
n2	hardcore techno, latin rap, southern rap, east coast rap	pies, grandmaster, hash, tricky, pimp
n3	pop-metal, melodic metal, detroit rock, flamenco guitar, math rock	roisin, pies, hash, dez, voulez
n4	new traditionalist, british folk-rock, progressive bluegrass, gabba, slowcore	csn, dez, voulez, shapes, daltrey
n5	rockabilly revival, new beat, progressive country, vocalion, freakbeat	hodges, precious, shanty, broonzy, dez
	$\chi^2\cdot$ LK variant	
n1	piano, orchestra, symphony, opera, violin	classical, symphony, composer, piano, orchestra
n2	rap, hip-hop, beats, dance, cool	hop, hip, rap, eminem, dj
n3	punk, metal, guitar, punk rock, hard	band, punk, metal, bands, live
n4	country, guitar, country music, folk, group	country, brooks, elvis, dylan, hits
n5	blues, jazz, country, soul	blues, jazz, willie, otis, john

Table 3.3: List of top ranked terms for nodes on the first level.

	with dictionary	without dictionary
Lagus & Kaski		
n5.1	blues, jazz, guitar, band, orchestra	blues, jazz, john, coltrane, basie
n5.2	soul, blues, song, gospel, pop	aretha, soul, redding, king, franklin
n5.3	reggae, ska, song, world, dancehall	marley, reggae, tosh, cliff, baez
n5.4	country, country music, song, bluegrass, folk	country, hank, elvis, cash, kenny
n5.5	band, song, pop, guitar, blues	elvis, roll, rate, band, bo
LK variant (with dictionary)		
n5.1	hot jazz, post-bop, vocalion, rondo, soul-jazz, classic jazz, hard bop, superstitious, octet	
n5.2	british blues, pornographic, colored, classic soul, sensual, erotic, precious, rap rock, stylish	
n5.3	vocal house, soca, british punk, gong, ragga, ska, dancehall, dancehall reggae, hard house	
n5.4	new traditionalist, yodelling, middle aged, country boogie, outlaw country, rockabilly revival	
n5.5	experimental rock, boogie rock, castanets, psychedelic pop, pagan, dream pop, crunchy	

Table 3.4: List of top ranked terms for the children of node n5.

3.3.5.2 Comparison of Term Selection Techniques

Table 3.3 lists all top-ranked words for the different approaches at the first level. Table 3.4 lists some examples of the second level (for the children of node n5). Comparing these tables with Figures 3.17 and 3.18 shows how different types of genres are described. For example, in most cases describing the classical cluster (node n1) works very well.

In general the following observations were made: First, the results using the dictionary are better in most cases. The difference is more obvious at the second level (for the children of node n5). In particular, using the dictionary avoids the frequent appearance of artist names.

Second, not all words in the domain specific dictionary make sense. Although not directly noticeable at the first level there are some words which appear frequently in the top-ranked words but do not convey much information: *world*, *uk*, *band*, *song*, *musical*. On the other hand, words such as *love* and *hate* are missing in the dictionary. Having a few meaningless words is not such a big problem. In an interactive interface the user could just click on the words to remove and the interface could be updated immediately. However, adding missing words to the dictionary requires scanning all the retrieved documents for occurrences.

Third, the performance of the non discriminating approaches ($tf \times idf$, LabelSOM) is very poor. On the other hand, all discriminative approaches (χ^2, LK, and combinations) yield interesting results with the dictionary. However, the LK variant by itself focuses too much on the differences. Obviously, to truly judge the quality of the different variations would require user studies. However, it seems that the approach from Lagus & Kaski performs slightly better than the others.

3.3.5.3 Discussion

One of the main problems is that the similarity measure relies on artist names. In many cases this name might have several meanings making it difficult to retrieve relevant web pages. Another problem is that many new and not so well known artist do not appear on web pages. Furthermore, the dictionary used contains terms mainly used in western culture. This limits the implemented approach to yesterday's mainstream western culture. However, the dictionary could easily be replaced. Another issue is the dynamics of web contents (e.g. [LG99]). This was studied in [KPW04] and [Kne04]. So far significant changes were observed in the Google ranks, but these did not have a significant impact on the similarity measure. In an ideal case, the web-based similarity measure would be complemented with data from other sources such market basket analysis, collaborative filtering, and audio signal analysis.

3.3.6 Conclusions

This section demonstrated possibilities to hierarchically organize music at the artist level. In particular, hierarchical overlapping clusters were described using a domain-specific dictionary. The results are very promising. However, so far no user based evaluation was conducted.

3.4 Dynamic Playlist Generation

Common approaches to creating playlists are to randomly shuffle a collection (e.g. iPod shuffle) or manually select songs. In this section heuristics to adapt playlists automatically given a song to start with (seed song) and immediate user feedback are presented and evaluated.

Instead of rich metadata audio-based similarity is used. The users give feedback by pressing a skip button if they dislike the current song. Songs similar to skipped songs are removed, while songs similar to accepted ones are added to the playlist. The heuristics are evaluated with hypothetical use cases. For each use case a specific user behavior (e.g. the user always skips songs by a particular artist) is assumed. The results show that using audio similarity and simple heuristics the number of necessary skips can be reduced drastically. This section describes the work presented in [PPW05a].

3.4.1 Introduction

There are different ways to create playlists. One extreme is to very carefully select each piece and the order in which the pieces are played. Another extreme is to randomly shuffle all pieces in a collection. While the first approach is very time consuming, the second approach produces useless results if the collection is very diverse.

This section describes an alternative which requires little user interaction even for very diverse collections. The goal is to minimize user input and maximize satisfaction. The *assumptions* are:

1. First, a seed song is given. The problem of browsing a large collection to find a song to start with is not addressed. (If more than one song to start with is given then the task is simplified.)

2. Furthermore, a skip button is available and easily accessible to the user. For example, this is the case if the user runs Winamp while browsing the Internet.

3. Finally, the assumption is made that the user is "lazy" and is willing to sacrifice quality for time. In particular, the assumption is that all the user is willing to do is press a skip button if the song currently played is a bad choice.

This section describes simple heuristics to dynamically propose the next song to be played in a playlist. The approach is based on audio similarity and takes the user's skipping behavior into account. The idea is to avoid songs similar to songs which were skipped and focus on songs similar to accepted ones. The heuristics are evaluated based on hypothetical use cases. These use cases are: (1) The user wants to listen to songs from the same genre as the seed song. (2) The user does not like an artist in this genre. (3) The user gets bored of the genre and wants to listen to a related genre. (The use cases are described in more detail later on.)

Structure of this Section

The remainder of this section is structured as follows. The next subsection reviews related work. Subsection 3.4.3 describes the heuristics to generate playlists and respond to user feedback. Subsection 3.4.4 described the evaluation procedure and the results. Subsection 3.4.5 describes the implementation used and gives some examples. Conclusions are drawn in Subsection 3.4.6.

3.4.2 Related Work

Previous work on playlist generation has partly dealt with algorithms to efficiently find a playlist which fulfills given constraints (e.g. [PRC00; AT01; AP02b; Pvd05]). These approaches assume the existence of rich metadata. A commercial product to generate playlists from proprietary metadata is available from Gracenote.[12] This "dynamic" playlist generator updates playlists when the music collection is modified.

A conceptually very similar approach to the approach described this section is the Internet radio station Last.FM[13] which creates a user profile based on immediate user feedback. Last.FM is built on collaborative filtering and uses "Love", "Skip", "Ban" buttons as input.

[12] http://www.gracenote.com/gn_products/playlist.html
[13] http://last.fm

Another approach using immediate user feedback and rich metadata was presented in [PE02]. The main difference to the approach in this section is that in the evaluations described in this section random shuffling would completely fail. Furthermore, the heuristics described here have no parameters which need to be trained and thus require less user feedback.

Unlike these previous approaches the approach in this section does not rely on metadata or collaborative filtering. Purely audio-based playlist generation was proposed for example in [Log02] and [PPW05c]. In [Log02] the author showed that simply using the n nearest songs to a seed song as a playlist performs relatively well. In [PPW05c] traveling salesman algorithms are used to find a path through the whole collection.

An interesting approach to playlist generation which partly uses audio-based similarity measures was presented by Goto & Goto [GG05]. The Musicream system they developed allows the user to playfully explore streams of music. Various features such as finding similar pieces, organizing and sorting playlists, and browsing experiences from the past are implemented. In contrast to the approach described here most features of the system assume a slightly more active user. Furthermore, while the assumption here is made that the playlists are generated on the users music collection, Musicream was primarily designed for "all-you-can-hear" music subscription services.

3.4.3 Method

This approach is based on an audio-based music similarity measure. Here a combination with 65% G30S, 15% FP, 5% FP Focus, and 15% FP Gravity is used (see Chapter 2). The details of the combination are described in [PFW05b] and [Pam05].

From a statistical (or machine learning) point of view it is problematic to use complex models (or learners) with a high number of parameters to learn user preferences. The main reason for this is that in an ideal case the number of negative examples is extremely low (e.g. less than 5) and even the number of positive examples is not very high (e.g. 20 tracks can fill an hour of music).

To generate the playlists the following 4 *heuristics* are used. Candidate songs are all songs in the collection which have not been played (or skipped) yet.

A. As suggested in [Log02] the n nearest neighbors to the seed song are played (n = accepted + skipped). This heuristic creates a static playlist and is the baseline.[14]

B. The candidate song closest to the last song accepted by the user is played. This is similar to heuristic A with the only difference that the

[14] A purely random playlist generator is not used as baseline because the generator would completely fail in the use cases described later on. The reason for this is the large number of genres in the collection.

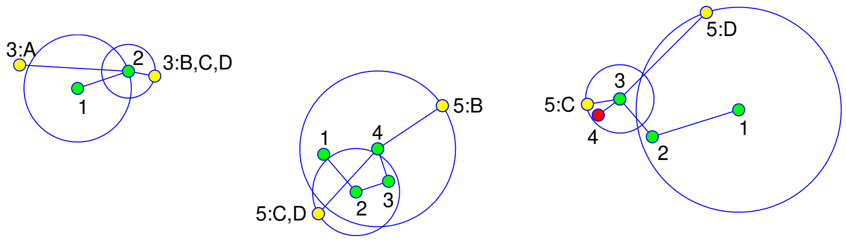


Figure 3.20: Illustration of the differences between the 4 heuristics.

seed song is always the last song accepted.

C. The candidate song closest to any of the accepted songs is played. Using the minimum distance for recommendations from song sets was proposed in [Log04].

D. For each candidate song, let d_a be the distance to the nearest accepted, and let d_s be the distance to the nearest skipped. If $d_a < d_s$, then add the candidate to the set S. From S play the song with smallest d_a. If S is empty, then play the candidate song which has the best (i.e. the lowest) d_a/d_s ratio.

Figure 3.20 shows the differences between the heuristics. On the left the first two songs were accepted by the user. There are two choices for the third song to be played. Heuristic A would chose the song on the left which is closer to the seed song, all other heuristics would chose the song closer to the second accepted song.

The case illustrated in the center of Figure 3.20 shows the difference between heuristics B on one side and C and D on the other side. In particular, the first 4 songs were accepted by the user. Heuristic B would select the song in the upper right which is closest to the last accepted song. On the other hand C and D would select the song in the lower left which is closer to song 2.

The third case illustrated on the right side of Figure 3.20 shows the difference between heuristics C and D. The first 3 songs were accepted by the user. The fourth song was rejected. Heuristic C ignores this and selects the song on the left which is very close to the rejected song. Heuristic D selects the song which is further away from the rejected song.

3.4.4 Evaluation

Evaluation of a playlist generation algorithm ultimately requires a user study. However, as shown in Chapter 2 the results from user ratings relate to the results from the much simpler to conduct genre-based evaluation. The approach described in this subsection is based on simulating user behavior and uses genre metadata.

3.4.4.1 Procedure (Hypothetical Use Cases)

In the hypothetical use cases the assumption is made that the user wants to listen to one hour of music which is approximately 20 songs. The number of skips are counted until these 20 songs are played. The use cases (UC) are the following:

UC-1. The user wants to listen to songs similar to the seed. This is measured by equating similarity with genre membership. Any song outside of the seed's genre is skipped. The evaluation is run using every song in the collection as seed.

UC-2. The user wants to listen to similar music but dislikes a particular artist (for not measurable reasons such as personal taste). To measure this the same approach as for UC-1 is used. An unwanted artist from the seed's genre (not the artist of the seed song) is randomly selected. Every time a song outside the seed's genre or from the unwanted artist is played, skip is pressed. The evaluation is run using every song in the collection as seed.

UC-3. The user's preferences change over time. This is measured as follows. Let A be the genre of the seed song and B a related genre which the user starts to prefer. The first 5 songs are accepted if they are from genre A. The next 10 are accepted if they are from either A or B. The last 5 are accepted if they are from B. Pairs of genres have been selected manually for this use case. The list of pairs can be found in Table 3.7. The evaluation is run using every song in genre A as seed. Unlike UC-1 and UC-2 it is possible that in UC-3 a state is reached where none of the candidate songs would be accepted although the number of accepted is less than 20. In such cases the remaining songs in the collection are added to the skip count.

One of the biggest problems for this evaluation is that there are not enough artists per genre to implement an artist filter. That is, playing several songs from the same artist right after each other is not avoided.

Another issue is that the assumption is made that only songs the user dislikes are skipped. However, if a song is skipped because, e.g., the user just heard it on the radio (but likes it otherwise) the heuristics will be mislead. To evaluate this some random skips could have been included. To solve this the user could be given more feedback options. For example, how long or hard the skip button is pressed could indicate how dissimilar the next song should be.

3.4.4.2 Data

The collection used contains 2522 tracks from 22 genres (see Table 3.5 for further statistics). It is the same collection described in Subsection 2.3.3 as DB-L. The difference is that this collection has not been cleaned. That is, there are a number

			Artists/Genre		Tracks/Genre	
Genres	Artists	Tracks	Min	Max	Min	Max
22	103	2522	3	6	45	259

Table 3.5: Statistics of the music collection.

	Heuristic	Min	Median	Mean	Max
UC-1	A	0	37.0	133.0	2053
	B	0	30.0	164.4	2152
	C	0	14.0	91.0	1298
	D	0	11.0	23.9	425
UC-2	A	0	52.0	174.0	2230
	B	0	36.0	241.1	2502
	C	0	17.0	116.9	1661
	D	0	15.0	32.9	453

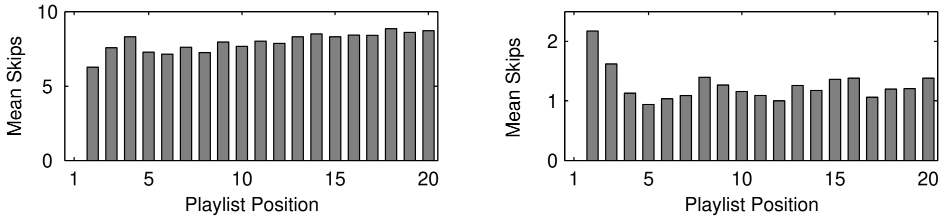
Table 3.6: Number of skips for UC-1 and UC-2.

of very short tracks which are not songs. (For example, introductions to an album etc.) The genres and the number of tracks per genre are listed in Fig. 3.22.

3.4.4.3 Results

For UC-1 using random shuffle to generate the playlist would require more than 300 skips in half of the cases while heuristic A requires less than 37 skips in half of the cases. Table 3.6 shows the results for UC-1 and UC-2. The main observation is that the performance increases from heuristic A to D. In general, there are a lot of outliers which is reflected in the large difference between mean and median. In a few cases almost all songs from the collection are proposed until 20 songs from the seed genre are in the playlist. Heuristic D has significantly fewer outliers. Half of all cases for heuristic D in UC-1 require less than 11 skips which might almost be acceptable.

Fig. 3.21 shows that for D/UC-1 there is a large number of skips after the



(a) Heuristic A (b) Heuristic D

Figure 3.21: Skips per playlist position for UC-1.

		AC	AJ	Blu	BN	Cel	DM	DnB	DT	Ele	ED	FR	GH	HC	HM	Ita	Jaz	JG	MM	Pun	Roc	Tra	T2	Sum
A Cappella	112		0.3	0.1	0.4	0.2		0.2	0.6	0.3	0.2	0.5	0.4	0.2	0.2	0.5	0.1	0.1	0.2	0.2	0.1	0.3	0.1	4.9
Acid Jazz	68	1.2		2.4	1.0	2.9	0.4	5.0	7.8	3.6	4.1	7.5	8.9	5.5	0.6	6.1	4.3	1.1	0.8	3.4	1.9	3.5	2.5	74.4
Blues	63	0.1	2.3		0.9	2.9	0.3	0.2	0.7	0.9	0.1	3.6	1.1	1.0	0.5	13.9	2.7		5.4	4.0	1.2	0.1	0.1	42.0
Bossa Nova	72	0.3	0.1	1.8		2.4				1.7	0.3	0.5	2.5		1.0	4.6	3.0	1.4					0.8	20.5
Celtic	132	0.8		1.6	0.5		0.7			0.2	0.1	1.1	0.1	0.3	0.3	3.9	0.2	0.2	1.0	1.3				12.5
Death Metal	72	0.3	0.2	0.3	0.2	0.9		0.8	1.2	0.9	0.3	5.3	0.6	0.3	5.2	0.7	0.2	0.2	2.5	7.5	0.1	0.6	0.2	28.3
DnB	72	0.3	1.2	0.2	0.1	0.1			1.0	0.3	2.7	3.2	3.1	10.0	0.7	1.7	1.8		0.1	1.9	2.6	1.2	0.8	32.9
Downtempo	124	1.6	4.2	2.9	2.5	3.7	0.6	4.0		5.8	4.0	6.2	8.6	4.7	2.8	3.3	3.0	1.7	2.2	4.7	1.8	3.1	2.8	74.4
Electronic	63	4.0	8.5	1.6	5.0	7.9	2.3	13.0	17.9		10.2	8.5	8.5	6.3	2.5	4.3	3.6	8.5	1.9	3.1	2.0	8.0	4.2	131.8
Euro−Dance	97	0.1	0.2	0.1	0.1	0.3	0.1	0.3	0.5	0.4		0.3	0.6	0.5	0.1	0.1	0.1		0.1	0.1	0.1	0.7	0.6	5.5
Folk−Rock	238	0.5	0.1	0.4	0.3	0.4	0.9	0.1	0.2	0.2	0.2		0.6	0.5	2.6	1.8	0.2	0.1	2.4	4.2	0.4	0.1	0.1	16.1
German Hip Hop	143	0.7	0.8	0.8	0.4	0.7		0.3	0.7	0.3	0.2	1.3		1.9	0.2	4.1	1.9	0.1	0.2	1.0	0.5	0.2	0.1	16.3
Hard Core Rap	164	0.4	1.6	1.2		0.5		0.8	0.2	0.3		3.2	8.3		0.1	4.9	4.0		0.2	2.7	3.2	0.1		31.9
Heavy Metal − Thrash	242					0.1	0.2		0.1	0.1		0.3	0.1			0.2	0.1		0.1	1.4				2.7
Italian	142	0.1	0.1	0.1	0.2	0.2	0.1	0.3	0.4	0.2	0.1	0.5	0.5	0.5	0.3		0.1	0.1	0.2	1.3	0.1	0.2		5.7
Jazz	64	0.5	5.9	3.6	4.4	2.6	0.1	0.3	3.3	0.3	0.6	2.5	12.3	7.6	0.7	9.2		2.2	1.2	1.8	1.7	0.2	0.2	61.1
Jazz Guitar	70				0.7												0.3							1.1
Melodic Metal	169	0.1	0.1	0.2	0.1	0.8	0.9	0.2	0.3	0.2	0.2	3.3	0.2	0.1	1.1	0.5	0.1			5.2		0.1	0.1	13.6
Punk	259	0.1	0.1			0.1	0.5	0.1	0.2	0.1		0.3	0.1	0.1	0.1	0.2	0.1		0.1			0.1		2.3
Reggae	47	1.2	1.0	1.7	0.3	1.5		0.8	0.7	0.9	0.7	1.1	5.0	10.2	0.2	3.7	1.2	0.6	0.1	0.6		0.2	0.2	32.1
Trance	64	0.7	2.6		0.5	1.3	0.9	2.3	2.5	2.9	16.1	4.1	3.1	1.4	1.0	2.3	0.6	0.7	2.0	4.0	0.4		4.2	53.6
Trance2	45	0.6	2.7	0.4	0.5	0.6	0.1	1.7	3.4	2.5	15.2	2.5	1.5	1.6	1.1	0.4	0.4	0.4	0.9	1.1	1.2	2.7		41.5

Figure 3.22: Mean number of skips per genre for heuristic D in UC-1. For example, the first line shows how many songs (in average, computed as the mean) from each genre were skipped for playlists starting with an a capella seed. The number to the left of the table (e.g. 112) is the number of total tracks in the genre. The number on the right side of the table (4.9) is the mean of the total number of skips.

Start	Goto	Heuristic A Median	Heuristic A Mean	Heuristic B Median	Heuristic B Mean	Heuristic C Median	Heuristic C Mean	Heuristic D Median	Heuristic D Mean
Euro-Dance	Trance	69.0	171.4	36.0	64.9	41.0	69.0	20.0	28.3
Trance	Euro-Dance	66.0	149.1	24.0	79.1	6.5	44.4	4.5	8.8
German Hip Hop	Hard Core Rap	33.0	61.9	32.0	45.6	31.0	40.7	23.0	28.1
Hard Core Rap	German Hip Hop	21.5	32.2	18.0	51.9	16.0	24.2	14.0	16.1
Heavy Metal/Thrash	Death Metal	98.5	146.4	54.0	92.5	58.0	61.1	28.0	28.4
Death Metal	Heavy Metal/Thrash	14.0	69.2	16.0	53.7	3.0	55.5	3.0	25.7
Bossa Nova	Jazz Guitar	68.5	228.1	32.0	118.7	54.0	61.1	22.0	21.3
Jazz Guitar	Bossa Nova	21.0	26.7	22.0	21.5	9.0	10.5	6.0	6.2
Jazz Guitar	Jazz	116.0	111.3	53.0	75.7	45.0	74.0	18.5	27.3
Jazz	Jazz Guitar	512.5	717.0	1286.0	1279.5	311.0	310.8	29.0	41.3
A Cappella	Death Metal	1235.0	1230.5	1523.0	1509.9	684.0	676.5	271.0	297
Death Metal	A Cappella	1688.0	1647.2	1696.0	1653.9	1186.0	1187.3	350.0	309.2

Table 3.7: Number of skips for UC-3.

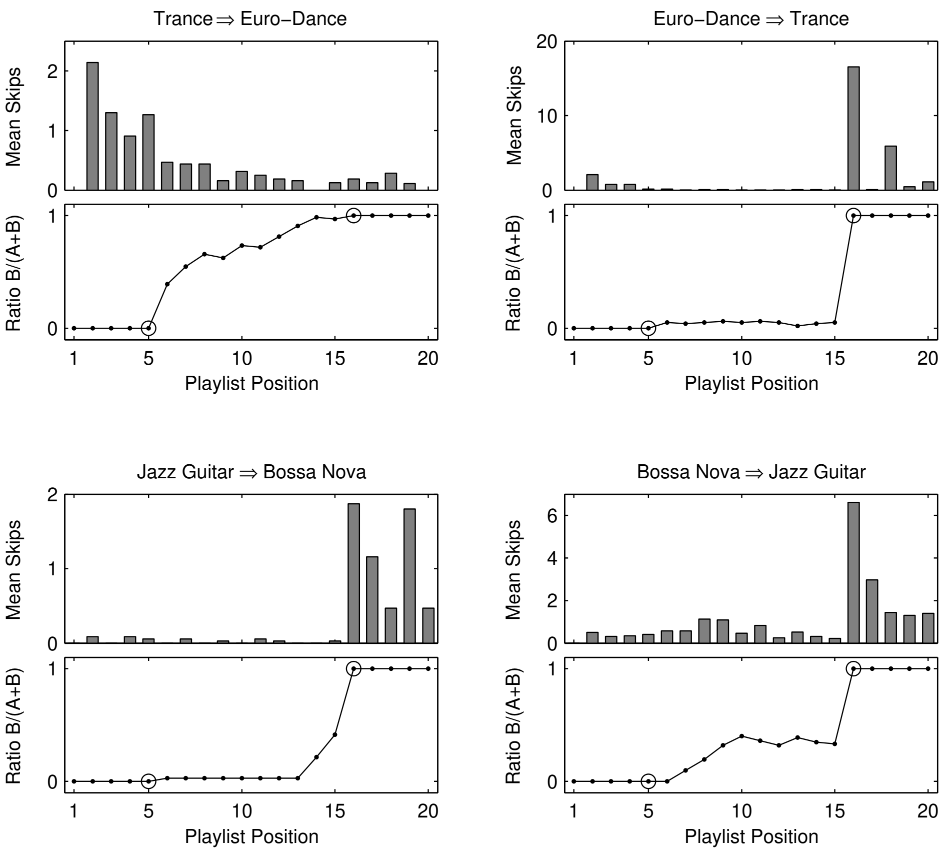


Figure 3.23: Average skips and genre ratio per playlist position for heuristic D in UC-3. The genre ratio is 0 if only genre A (the genre of the seed) is played and 1 if only genre B (destination genre) is played. The circle marks the last and first song which is forced to be from a specific genre.

first song (seed song). Once the system has a few positive examples the number of skips decreases. On the other hand, for heuristic A, the number of skips gradually increases with the playlist position. (Note that one must be careful when interpreting the mean because it is strongly influenced by a few outliers.)

Fig. 3.22 shows that for D/UC-1 some genres work very well (e.g. jazz guitar or heavy metal - trash), while others fail (e.g. electronic or downtempo). However, some of the failures make sense. For example, before 20 pieces from electronic are played, in average almost 18 pieces from downtempo are proposed.

Table 3.7 gives the results for UC-3. As for the other use cases the performance increases from A to D in most cases. The pair a capella to death metal was included as an extreme to show the limitations (such a transition it is not considered to be a likely user scenario). In three of the cases for heuristic D the median seems to be acceptably low. However, using an artist filter these results would surely be significantly worse.

The number of skips depends a lot on the direction of the transition. For example, moving from jazz guitar to bossa nova requires, in half of the cases, less than 6 skips. Moving in the other direction requires almost 3 times as many skips. This is also reflected in Fig. 3.22. Specifically, jazz guitar to bossa nova works well because jazz guitar is mainly confused with bossa nova. On the other hand bossa nova is confused with many other genres. The same can be observed, e.g., for the pair trance and euro-dance.

Fig. 3.23 shows where skips occur for UC-3 and heuristic D, and how often each genre was played per playlist position. In some cases during the transition phase (where genre A or B are accepted) basically only genre A is played. When the transition is enforced (after the 15th song in the playlist) the number of skips drastically increases. In other cases the transition works very nicely. An obvious direction for further improvement is to include a memory effect to allow the system to forget previous user choices. However, preliminary experiments conducted in this direction did not show significant improvements.

3.4.5 Implementation and Examples

To make the results more tangible a Matlab based interface was implemented (see Figure 3.24). On of the first findings was that some form of artist filter needed to be implemented to generate interesting playlists. An example is shown in Table 3.8. The seed song was "When the Saints Go Marchin In". The next songs in the list are all from the same artist. Table 3.9 shows the effects of an simple artist filter using the same seed. In particular, the filter enforces that the next n songs cannot be from the same artist. Here n was set to 3 (AF=3).

Tables 3.10 and 3.11 show how the user feedback changes the playlists. In case of Table 3.10 the feedback leads to a clear improvment. In the case of Table 3.11 the system tries mixing a different genre into the playlist every time. First melodic metal is mixed into the list. After melodic metal is rejected death metal is mixed

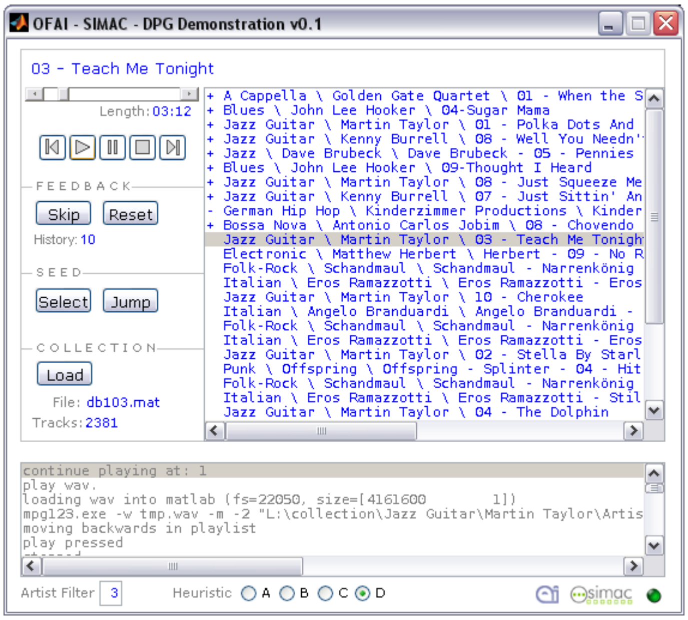


Figure 3.24: Screenshot of the Matlab user interface. The skip button is located on the center left. The heuristics can be adjusted in the lower center, the artist filter can be set lower left. A "+" in front of a song marks an accepted song, a "–" marks a reject. The two main ways to start a playlist are either to randomly jump around in the collection until an acceptable seed is found, or to manually select a particular song.

+	A Cappella / Golden Gate Quartet / When the Saints Go Marching In
	A Cappella / Golden Gate Quartet / I heard Zion moan
	A Cappella / Golden Gate Quartet / Noah
	A Cappella / Golden Gate Quartet / Dipsy Doodle
	A Cappella / Golden Gate Quartet / Sampson
	A Cappella / Golden Gate Quartet / Lead me on and on
	...

Table 3.8: Example playlist where no artist filter was used.

+	A Cappella / Golden Gate Quartet / When the Saints Go Marching In
	Blues / John Lee Hooker / Sugar Mama
	Jazz Guitar / Martin Taylor / Polka Dots And Moonbeams
	Jazz Guitar / Kenny Burrell / Well You Needn't
	Jazz / Dave Brubeck / Pennies From Heaven
	Blues / John Lee Hooker / Thought I Heard
	...

Table 3.9: Example where an artist filter was used (AF=3) with heuristic D.

+	Acid Jazz / Jazzanova / Introspection
	Punk / Rancid / Life Won't Wait - Life Won't Wait
	Italian / Eros Ramazzotti / Dove c'è Musica - Yo sin Ti
	...
+	Acid Jazz / Jazzanova / Introspection
−	Punk / Rancid / Life Won't Wait - Life Won't Wait
	Electronic / Kaito / Scene
	...

Table 3.10: Example how user feedback improves the playlist using heuristic D and AF=3. The "−" marks a rejected song. The "+" marks an accepted song.

into the list. After death metal is rejected heavy metal is mixed into the list. At least it is obvious that the system is not repeating the same mistakes.

3.4.6 Conclusions

This section described an approach to dynamically create playlists based on the user's skipping behavior. The approach was evaluated using hypothetical use cases for which specific behavior patterns were assumed. Compared to the approach suggested in [Log02], heuristic D reduces the number of skips drastically. In some of the cases the necessary number of skips seems low enough for a real world application. The main limitation of the evaluation is that no artist filter was used (to avoid having a large number of pieces from the same artist right after each other in a playlist). A filter was not implemented due to the small number of artist per genre.

The heuristic depends most of all on the similarity measure. Any improvements would lead to fewer skips. However, implementing memory effects (to forget past decisions of the user) or allowing the similarity measure to adapt to the user's behavior are also interesting directions. For use cases related to changing user preferences a key issue might be to track the direction of this change. Incorporating additional information such as web-based artist similarity or modeling the user's context more accurately (based on data from long term usage) are other options.

Although evaluations based on hypothetical use cases seems to be sufficient for the current development state, experiments with human listeners will be necessary in the long run. However, this would require implementing a number of additional functions. For example, it is necessary to allow the users to adjust how much variance they desire. Basically the question is how many songs should be played until the same pieces or different pieces from the same artists are repeated. In addition, different types of skips, tools to manage the user's feedback history, and tools to manage different radio stations (similar to interface implemented by pandora.com) are desirable functionalities.

+	Punk / Bad Religion / The Empire Strikes First - The Quickening
+	Punk / Green Day / Nimrod - Haushinka
	Melodic Metal / Nightwish / Once - Wish I Had an Angel
	Melodic Metal / Stratovarious / Elements Part II - Dreamweaver
	... (the next 10 include 2 melodic metal, 1 death metal and 7 punk songs)
+	Punk / Bad Religion / The Empire Strikes First - The Quickening
+	Punk / Green Day / Nimrod - Haushinka
−	Melodic Metal / Nightwish / Once - Wish I Had an Angel
	Death Metal / Borknagar / Epic - Cyclus
	Punk / Bad Religion / The Empire Strikes First - All There Is
	... (the next 10 include 2 death metal and 8 punk songs)
+	Punk / Bad Religion / The Empire Strikes First - The Quickening
+	Punk / Green Day / Nimrod - Haushinka
−	Melodic Metal / Nightwish / Once - Wish I Had an Angel
−	Death Metal / Borknagar / Epic - Cyclus
	Punk / Bad Religion / The Empire Strikes First - All There Is
	Punk / Green Day / Nimrod - Jinx
	Punk / Rancid / Life Won't Wait - 1998
	... (the next 10 include 3 heavy metal and 1 Italian and 7 punk songs)

Table 3.11: Example how user feedback effects the playlist using heuristic D and AF=3.

3.5 Conclusions

In this chapter 3 different applications of music similarity measures were described. The first application demonstrated how a metaphor of geographic maps can be used as interface which supports exploration of music collections. The interface allows the user to browse different views. Each view is defined by a specific aspect of similarity (or combinations of different aspects). The second application demonstrated how web-based similarity can be used to hierarchically organize a music collection at the artist level into overlapping groups. In addition, each group was summarized to allow the user to easily identify interesting groups without requiring the user to know any of the artists' names. The third application demonstrated how minimal user interaction can be used to generate playlists. In particular, based on the user's skipping behavior the playlist is dynamically updated.

Each of these applications has its limitations. The primary limitation is the performance of the similarity measure. However, each application deals differently with this limitation. The Islands of Music application gives the user the responsibility of fine tuning the similarity measure. This would work if the aspects of similarity the user is asked to mix were more meaningful. The overlap in the hierarchical organization increased the number of pieces considered similar to any given piece and thus avoids omitting similar pieces at the cost of presenting the user more pieces than necessary. However, the main advantage is that the user is not given a wrong impression about the accuracy of the system. For playlist generation a certain amount of variance is very desirable. A perfect similarity measure is not necessary to create interesting results. However, it would be important to improve the similarity measures to the point that no unacceptable matches are returned.

Future work includes combining different approaches (for example, audio and web-based similarity). Furthermore, bringing applications to a level which allows user studies is highly desirable. Finally, in the long run integrating functionalities in currently used tools (e.g. developing iTunes or Winamp plugins) would allow realistic tests of functionalities and seems promising in some cases.

Chapter 4
Conclusions

In this thesis computational models of music similarity and their application were described. Chapter 2 gave an introduction to similarity measures. A thorough evaluation of state-of-the-art techniques was presented. Particular precautions were taken to avoid overfitting. The evaluation procedures, and especially the use of an artist filter, are highly recommended for future evaluations. Furthermore, the results of the presented listening test justify the use of genre classification based evaluations for similarity measures (in particular, to efficiently evaluate large parameter spaces). One of the outcomes of Chapter 2 is a combined similarity measure which performs significantly better than the baseline technique on most collections and exhibits fewer anomalies in the similarity space.

Chapter 3 described three applications of similarity measures. These applications demonstrated how music collections can be visualized, organized hierarchically, summarized with words found on web pages, and how playlists can be generated with minimum user interaction.

Outlook and Future Work

Some of the techniques described in this thesis are ready for deployment. Audio-based similarity measures produce acceptable results in most cases. As add-on functionality they can be integrated in various applications such as media players. Specifically, the dynamic playlist generation application described in Chapter 3 is almost at the point where its integration into common media players makes sense.

Nevertheless, a large number of research questions remain open. An obvious direction for future work is to improve the similarity measure. One option is to improve the features. For example, the "Noisiness" feature described in Chapter 2 can easily be improved. In addition, there are a large number of features which could be tried instead of the ones used in this thesis. In the long run replacing the low-level audio statistics with high-level features is very desirable.

Another option to improve the similarity measure is to improve the combination. The linear combination used in this thesis is surely not the best solution. Furthermore, an important direction is the further reduction of computation times and the development of techniques to deal with music collections containing millions of pieces.

Interesting directions for future work include alternative approaches to apply computational models of music similarity. In addition, each of the three applications described in Chapter 3 offers a lot of room for improvements. A particularly interesting direction is the combination of information from different sources (e.g. audio-based analysis combined with information from the web). Finally, user studies are needed to verify various assumptions made about user requirements and behavior.

Bibliography

[AHH+03] Eric Allamanche, Jürgen Herre, Oliver Hellmuth, Thorsten Kastner, and Christian Ertel, *A Multiple Feature Model for Musical Similarity Retrieval*, Proceedings of the ISMIR International Conference on Music Information Retrieval (Baltimore, MD), 2003, pp. 217–218.

[AP02a] Jean-Julien Aucouturier and François Pachet, *Music Similarity Measures: What's the Use?*, Proceedings of the ISMIR International Conference on Music Information Retrieval (Paris, France), 2002, pp. 157–163.

[AP02b] ———, *Scaling up music playlist generation*, Proceedings of the IEEE International Conference on Multimedia Expo (Lausanne, Switzerland), 2002.

[AP03] ———, *Representing Musical Genre: A State of the Art*, Journal of New Music Research **32** (2003), no. 1, 83–93.

[AP04a] ———, *Improving Timbre Similarity: How high is the sky?*, Journal of Negative Results in Speech and Audio Sciences **1** (2004), no. 1, http://journal.speech.cs.cmu.edu/articles/2004/3.

[AP04b] ———, *Tools and Architecture for the Evaluation of Similarity Measures: Case Study of Timbre Similarity*, Proceedings of the ISMIR International Conference on Music Information Retrieval (Barcelona, Spain), 2004, pp. 198–203.

[AT01] Masoud Alghoniemy and Ahmed H. Tewfik, *A network flow model for playlist generation*, Proceedings of the IEEE International Conference Multimedia and Expo (Tokyo, Japan), 2001.

[Bau05] Stephan Baumann, *Artificial Listening Systems: Modellierung und Approximation der subjektiven Perzeption von Musikähnlichkeit (Models for the approximation of the subjective perception of music similarity)*, Ph.D. thesis, Technical University of Kaiserslautern, Germany, 2005.

[BCE05] James Bergstra, Norman Casagrande, and Douglas Eck, *Genre Classification: Timbre- and Rhythm-Based Multiresolution Audio Classification*, Proceedings of the MIREX Annual Music Information Retrieval eXchange (London), 2005.

[BDA+05] Juan Pablo Bello, Laurent Daudet, Samer Abdallah, Chris Duxbury, Mike Davies, and Mark B. Sandler, *A Tutorial on Onset Detection in Music Signals*, IEEE Transactions on Speech and Audio Processing **13** (2005), no. 5, 1035–1047.

[BEL03] Adam Berenzweig, Dan P.W. Ellis, and Steve Lawrence, *Anchor Space for Classification and Similarity Measurement of Music*, Proceedings of the IEEE International Conference on Multimedia and Expo (Baltimore, MD), 2003, pp. 29–32.

[BH03] Stephan Baumann and Oliver Hummel, *Using Cultural Metadata for Artist Recommendation*, Proceedings of the WedelMusic Conference (Leeds, UK), 2003, pp. 138–141.

[BH04] Stephan Baumann and John Halloran, *An Ecological Approach to Multimodal Subjective Music Similarity Perception*, Proceedings of the Conference on Interdisciplinary Musicology (Graz, Austria), 2004.

[Bis95] Christopher M. Bishop, *Neural Networks for Pattern Recognition*, Oxford University Press, Oxford, 1995.

[BLEW03] Adam Berenzweig, Beth Logan, Daniel P.W. Ellis, and Brian Whitman, *A Large-Scale Evaluation of Acoustic and Subjective Music Similarity Measures*, Proceedings of the ISMIR International Conference on Music Information Retrieval (Baltimore, MD), 2003, pp. 99–105.

[BM01] Ella Bingham and Heikki Mannila, *Random Projection in Dimensionality Reduction: Applications to Image and Text Data*, Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2001, pp. 245–250.

[BMCM95] Frédéric Bimbot, Ivan Magrin-Chagnolleau, and Luc Mathan, *Second-Order Statistical Measures for Text-Independent Speaker Identification*, Speech Communication **17** (1995), no. 1-2, 177–192.

[BP03] Stephan Baumann and Tim Pohle, *A Comparison of Music Similarity Measures for a P2P Application*, Proceedings of the International Conference on Digital Audio Effects (London), 2003, pp. 285–288.

[BP05] Juan P. Bello and Jeremy Pickens, *A Robust Mid-level Representation for Harmonic Content in Music Signals*, Proceedings of the ISMIR International Conference on Music Information Retrieval (London), 2005, pp. 304–311.

[BPS04] Stephan Baumann, Tim Pohle, and Vembu Shankar, *Towards a Sociocultural Compatibility of MIR Systems*, Proceedings of the ISMIR International Conference on Music Information Retrieval (Barcelona, Spain), 2004, pp. 460–465.

[BSW98] Christopher M. Bishop, Markus Svensén, and Christopher K. I. Williams, *GTM: The Generative Topographic Mapping*, Neural Computation **10** (1998), no. 1, 215–234.

[BWD02] Kevin W. Boyack, Brian N. Wylie, and George S. Davidson, *Domain Visualization Using VxInsight for Science and Technology Management*, Journal of the American Society for Information Science and Technology **53** (2002), no. 9, 764–774.

[CEK05] Norman Casagrande, Douglas Eck, and Balzs Kgl, *Frame-Level Audio Feature Extraction Using AdaBoost*, Proceedings of the ISMIR International Conference on Music Information Retrieval (London), 2005, pp. 345–350.

[CKGB02] Pedro Cano, Martin Kaltenbrunner, Fabien Gouyon, and Eloi Batlle, *On the Use of FastMap for Audio Retrieval and Browsing*, Proceedings of the ISMIR International Conference on Music Information Retrieval (Paris, France), 2002, pp. 275–276.

[CKW+05] Pedro Cano, Markus Koppenberger, Nicolas Wack, José Pedro Garcia, Jaume Masip, Óscar Celma, David García, Emilia Gómez, Fabien Gouyon, Enric Guaus, Perfecto Herrera, Jordi Masseguer, Bee Suan Ong, Miguel Ramirez, Sebastian Streich, and Xavier Serra, *An Industrial-Strength Content-based Music Recommendation System*, Proceedings of 28th Annual International ACM SIGIR Conference (Salvador, Brazil), 2005, p. 673.

[CPL94] Piero Cosi, Giovanni De Poli, and Giampaolo Lauzzana, *Auditory Modeling and Self-Organizing Neural Networks for Timbre Classification*, Journal of New Music Research **23** (1994), 71–98.

[CPZ97] Paolo Ciaccia, Marco Patella, and Pavel Zezula, *M-tree: An Efficient Access Method for Similarity Search in Metric Spaces*, Proceedings of the International Conference on Very Large Data Bases (Athens, Greece), 1997, pp. 426–435.

[CRH05] Òscar Celma, Miguel Ramírez, and Perfecto Herrera, *Foafing the music: A music recommendation system based on RSS feeds and user preferences*, Proceedings of the ISMIR International Conference on Music Information Retrieval (London), 2005, pp. 464–467.

[DFT04] J. Stephen Downie, Joe Futrelle, and David Tcheng, *The International Music Information Retrieval Systems Evaluation Laboratory: Governance, Access and Security*, Proceedings of the ISMIR International Conference on Music Information Retrieval (Barcelona, Spain), 2004, pp. 9–14.

[DGW04] Simon Dixon, Fabien Gouyon, and Gerhard Widmer, *Towards Characterisation of Music via Rhythmic Patterns*, Proceedings of the ISMIR International Conference on Music Information Retrieval (Barcelona, Spain), 2004, pp. 509–516.

[DMR00] Michael Dittenbach, Dieter Merkl, and Andreas Rauber, *The Growing Hierarchical Self-Organizing Map*, Proceedings of the International Joint Conference on Neural Networks (Como, Italy), vol. VI, IEEE Computer Society, 2000, pp. 15–19.

[DPW03] Simon Dixon, Elias Pampalk, and Gerhard Widmer, *Classification of Dance Music by Periodicity Patterns*, Proceedings of the ISMIR International Conference on Music Information Retrieval (Baltimore, MD, USA), 2003, pp. 159–166.

[EWBL02] Daniel Ellis, Brian Whitman, Adam Berenzweig, and Steve Lawrence, *The Quest For Ground Truth in Musical Artist Similarity*, Proceedings of the ISMIR International Conference on Music Information Retrieval (Paris, France), 2002.

[Fas82] Hugo Fastl, *Fluctuation Strength and Temporal Masking Patterns of Amplitude-Modulated Broad-Band Noise*, Hearing Research **8** (1982), 59–69.

[FB01] Mikael Fernström and Eoin Brazil, *Sonic Browsing: An Auditory Tool for Multimedia Asset Management*, Proceedings of the International Conference on Auditory Display (Espoo, Finland), 2001, pp. 132–135.

[FG94] Bernhard Feiten and Stefan Günzel, *Automatic Indexing of a Sound Database Using Self-Organizing Neural Nets*, Computer Music Journal **18** (1994), no. 3, 53–65.

[Foo97] Jonathan T. Foote, *Content-Based Retrieval of Music and Audio*, Proceedings of SPIE Multimedia Storage and Archiving Systems II (Bellingham, WA), vol. 3229, SPIE, 1997, pp. 138–147.

[Foo99] Jonathan T. Foote, *Visualizing Music and Audio using Self-Similarity*, Proceedings of ACM Multimedia '99 (Orlando, Florida, USA), October 1999, pp. 77–80.

[FR01] Markus Frühwirth and Andreas Rauber, *Self-Organizing Maps for Content-Based Music Clustering*, Proceedings of the Twelfth Italian Workshop on Neural Nets (Vietri sul Mare, Salerno, Italy), IIAS, 2001.

[Frü01] Markus Frühwirth, *Automatische Analyse und Organisation von Musikarchiven (Automatic Analysis and Organization of Music Archives)*, Master's thesis, Vienna University of Technology, Austria, 2001.

[GB05] Emilia Gómez and Jordi Bonada, *Tonality visualization of polyphonic audio*, Proceedings of International Computer Music Conference 2005 (Barcelona, Spain), 2005, pp. 57–60.

[GDPW04] Fabien Gouyon, Simon Dixon, Elias Pampalk, and Gerhard Widmer, *Evaluating rhythmic descriptors for musical genre classification*, Proceedings of the AES 25th International Conference (London, UK), 2004.

[GG05] Masataka Goto and Takayuki Goto, *Musicream: New Music Playback Interfaces for Streaming, Sticking, Sorting, and Recalling Musical Pieces*, Proceedings of the ISMIR International Conference on Music Information Retrieval (London), 2005, pp. 404–411.

[GHNO03] Masataka Goto, Hiroki Hashiguchi, Takuichi Nishimura, and Ryuichi Oka, *RWC Music Database: Music Genre Database and Musical Instrument Sound Database*, Proceedings of the ISMIR International Conference on Music Information Retrieval (Baltimore, MD), 2003, pp. 229–230.

[GLCS95] Asif Ghias, Jonathan Logan, David Chamberlin, and Brian C. Smith, *Query by Humming: Musical Information Retrieval in an Audio Database*, Proceedings of the 3rd ACM International Conference on Multimedia (San Fancisco, CA), 1995, pp. 231–236.

[Got03] Masataka Goto, *A Chorus-Section Detecting Method for Musical Audio Signals*, Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing vol. V, 2003, pp. 437–440.

[Gou05] Fabien Gouyon, *A computational approach to rhythm description — Audio features for the computation of rhythm periodicity functions and their use in tempo induction and music content processing*, Ph.D. thesis, Music Technology Group, Universitat Pompeu Fabra, Barcelona, Spain, 2005.

[GPW04] Werner Goebl, Elias Pampalk, and Gerhard Widmer, *Exploring Expressive Performance Trajectories: Six Famous Pianists Play Six Chopin Pieces*, Proceedings of the 8th International Conference on Music Perception and Cognition (ICMPC8) (Adelaide, Australia), 2004, pp. 505–509.

[Har96] William H. Hartmann, *Signals, Sound, and Sensation*, AIP Press, 1996.

[HSG04] Perfecto Herrera, Vegard Sandvold, and Fabien Gouyon, *Percussion-Related Semantic Descriptors of Music Audio Files*, Proceedings of the AES 25th International Conference (London, UK), 2004.

[Jeh05] Tristan Jehan, *Hierarchical Multi-class Self Similarities*, Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio Acoustics (New Paltz, NY), 2005.

[Kas98] Samuel Kaski, *Dimensionality reduction by random mapping*, Proceedings of the International Joint Conference on Neural Networks, vol. 1, 1998, pp. 413–418.

[Kas99] ——, *Fast Winner Search for SOM-Based Monitoring and Retrieval of High-Dimensional Data*, Proceedings of the Ninth International Conference on Artificial Neural Networks (London), vol. 2, 1999, pp. 940–945.

[KHLK98] Samuel Kaski, Timo Honkela, Krista Lagus, and Teuvo Kohonen, *WEBSOM – Self-Organizing Maps of Document Collections*, Neurocomputing **21** (1998), 101–117.

[KL05] David Kusek and Gerd Leonhard, *The Future of Music*, Berklee Press, 2005.

[Kne04] Peter Knees, *Automatische Klassifikation von Musikkünstlern basierend auf Web-Daten (Automatic Classification of Music Artists Based on Web-Data)*, Master's thesis, Vienna University of Technology, 2004.

[KO90] Pasi Koikkalainen and Erkki Oja, *Self-Organizing Hierarchical Feature Maps*, Proceedings of the International Joint Conference on Neural Networks (San Diego, CA), 1990.

[Koh82] Teuvo Kohonen, *Self-Organizing Formation of Topologically Correct Feature Maps*, Biological Cybernetics **43** (1982), 59–69.

[Koh01] ——, *Self-Organizing Maps*, 3rd ed., Springer Series in Information Sciences, vol. 30, Springer, Berlin, 2001.

[KPW04] Peter Knees, Elias Pampalk, and Gerhard Widmer, *Artist Classification with Web-based Data*, Proceedings of the ISMIR International Conference on Music Information Retrieval (Barcelona, Spain), 2004, pp. 517–524.

[KW78] Joseph B. Kruskal and Myron Wish, *Multidimensional Scaling*, Paper Series on Quantitative Applications in the Social Sciences, no. 07-011, Sage Publications, Newbury Park, CA, 1978.

[LEB03] Beth Logan, Daniel P.W. Ellis, and Adam Berenzweig, *Toward Evaluation Techniques for Music Similarity*, Proceedings of the SIGIR Workshop on the Evaluation of Music Information Retrieval Systems (Toronto, Canada), 2003.

[LG99] Steve Lawrence and C. Lee Giles, *Accessibility of Information on the Web*, Nature **400** (1999), no. 6740, 107–109.

[LK99] Krista Lagus and Samuel Kaski, *Keyword Selection Method for Characterizing Text Document Maps*, Proceedings of ICANN99, Ninth International Conference on Artificial Neural Networks (London), vol. 1, IEE, 1999, pp. 371–376.

[LKM04] Beth Logan, Andrew Kositsky, and Pedro Moreno, *Semantic Analysis of Song Lyrics*, Proceedings of IEEE International Conference on Multimedia and Expo 2004 (Taipei, Taiwan), 2004.

[Log00] Beth Logan, *Mel Frequency Cepstral Coefficients for Music Modeling*, Proceedings of the ISMIR International Symposium on Music Information Retrieval (Plymouth, MA), 2000.

[Log02] ———, *Content-Based Playlist Generation: Exploratory Experiments*, Proceedings of the ISMIR International Conference on Music Information Retrieval (Paris, France), 2002, pp. 295–296.

[Log04] ———, *Music Recommendation from Song Sets*, Proceedings of the ISMIR International Conference on Music Information Retrieval (Barcelona, Spain), 2004, pp. 425–428.

[LOL03] Tao Li, Mitsunori Ogihara, and Qi Li, *A Comparative Study on Content-based Music Genre Classification*, Proceedings of the ACM SIGIR International Conference on Research and Development in Informaion Retrieval (Toronto, Canada), 2003, pp. 282–289.

[LR05] Thomas Lidy and Andreas Rauber, *Evaluation of Feature Extractors and Psycho-Acoustic Transformations for Music Genre Classification*, Proceedings of the ISMIR International Conference on Music Information Retrieval (London), 2005, pp. 34–41.

[LS01] Beth Logan and Ariel Salomon, *A Music Similarity Function Based on Signal Analysis*, Proceedings of the IEEE International Conference on Multimedia and Expo (Tokyo, Japan), 2001.

[Lüb05] Dominik Lübbers, *SoniXplorer: Combining Visualization and Auralization for Content-Based Exploration of Music Collections*, Proceedings of the ISMIR International Conference on Music Information Retrieval (London), 2005, pp. 590–593.

[Mac67] J. MacQueen, *Some Methods for Classification and Analysis of Multivariate Observations*, Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability (Berkeley and Los Angeles, CA) (L. M. Le Cam and J. Neyman, eds.), Statistics, vol. I, University of California Press, 1967, pp. 281–297.

[MB03] Martin F. McKinney and Jeroen Breebaart, *Features for Audio and Music Classification*, Proceedings of the ISMIR International Conference on Music Information Retrieval (Baltimore, MD), 2003, pp. 151–158.

[ME05] Michael Mandel and Dan Ellis, *Song-Level Features and Support Vector Machines for Music Classification*, Proceedings of the ISMIR International Conference on Music Information Retrieval (London), 2005, pp. 594–599.

[MHV03] Pedro J. Moreno, Purdy P. Ho, and Nuno Vasconcelos, *A Kullback-Leibler Divergence Based Kernel for SVM Classification in Multimedia Applications*, Proceedings of Neural Information Processing Systems (Vancouver, Canada), 2003.

[Mii90] Risto Miikkulainen, *Script Recognition with Hierarchical Feature Maps*, Connection Science **2** (1990), 83–101.

[MKC05] Meinard Müller, Frank Kurth, and Michael Clausen, *Audio Matching via Chroma-Based Statistical Features*, Proceedings of the ISMIR International Conference on Music Information Retrieval (London), 2005, pp. 288–295.

[Moe02] Dirk Moelants, *Preferred Tempo Reconsidered*, Proceedings of the 7th International Conference on Music Perception and Cognition (Sydney), 2002, pp. 580–583.

[MR00] Dieter Merkl and Andreas Rauber, *Document Classification with Unsupervised Neural Networks*, Soft Computing in Information Retrieval (F. Crestani and G. Pasi, eds.), Physica Verlag, 2000, pp. 102–121.

[MST05] Anders Meng and John Shawe-Taylor, *An Investigation of Feature Models for Music Genre Classification Using the Support Vector Classifier*, Proceedings of the ISMIR International Conference on Music Information Retrieval (London), 2005, pp. 604–609.

[MUNS05] Fabian Mörchen, Alfred Ultsch, Mario Nöcker, and Christian Stamm, *Databionic Visualization of Music Collections According to Perceptual Distance*, Proceedings of the ISMIR International Conference on Music Information Retrieval (London), 2005, pp. 396–403.

[Nab01] Ian Nabney, *Netlab: Algorithms for Pattern Recognition*, Advances in Pattern Recognition, Springer, Berlin, 2001.

[NDR05] Robert Neumayer, Michael Dittenbach, and Andreas Rauber, *PlaySOM and PocketSOMPlayer, Alternative Interfaces to Large Music Collections*, Proceedings of the ISMIR International Conference on Music Information Retrieval (London), 2005, pp. 618–623.

[OH04] Bee-Suan Ong and Perfecto Herrera, *Computing Structural Descriptions of Music through the Identification of Representative Excerpts from Audio Files*, Proceedings of 25th International AES Conference (London), 2004.

[Opp69] Alan V. Oppenheim, *A speech analysis-synthesis system based on homomorphic filtering*, Journal of the Acostical Society of America **45** (1969), 458–465.

[Pam01] Elias Pampalk, *Islands of Music: Analysis, Organization, and Visualization of Music Archives*, Master's thesis, Vienna University of Technology, Department of Software Technology and Interactive Systems, 2001.

[Pam03] ______, *Aligned Self-Organizing Maps*, Proceedings of the Workshop on Self-Organizing Maps (WSOM'03) (Kitakyushu, Japan), Kyushu Institute of Technology, 2003, pp. 185–190.

[Pam04] Elias Pampalk, *A Matlab Toolbox to Compute Music Similarity From Audio*, Proceedings of the ISMIR International Conference on Music Information Retrieval (Barcelona, Spain), 2004, pp. 254–257.

[Pam05] Elias Pampalk, *Speeding Up Music Similarity*, Proceedings of the MIREX Annual Music Information Retrieval eXchange (London), 2005.

[PC00] François Pachet and Daniel Cazaly, *A Taxonomy of Musical Genres*, Proceedings of RIAO 2000 Content-Based Multimedia Information Access (Paris, France), 2000.

[PDW03a] Elias Pampalk, Simon Dixon, and Gerhard Widmer, *Exploring Music Collections by Browsing Different Views*, Proceedings of the ISMIR International Conference on Music Information Retrieval (Baltimore, MD), John Hopkins University, 2003, pp. 201–208.

[PDW03b] ______, *On the Evaluation of Perceptual Similarity Measures for Music*, Proceedings of the Sixth International Conference on Digital Audio Effects (DAFx-03) (London, UK), 2003, pp. 7–12.

[PDW04] ______, *Exploring Music Collections by Browsing Different Views*, Computer Music Journal **28** (2004), no. 2, 49–62.

[PE02] Steffen Pauws and Berry Eggen, *PATS: Realization and User Evaluation of an Automatic Playlist Generator*, Proceedings of the ISMIR International Conference on Music Information Retrieval (Paris, France), 2002.

[PFW05a] Elias Pampalk, Arthur Flexer, and Gerhard Widmer, *Hierarchical Organization and Description of Music Collections at the Artist Level*, Proceedings of the 9th European Conference on Research and Advanced Technology for Digital Libraries (Vienna, Austria), 2005, pp. 37–48.

[PFW05b] ______, *Improvements of Audio-Based Music Similarity and Genre Classification*, Proceedings of the ISMIR International Conference on Music Information Retrieval (London), 2005, pp. 628–633.

[PGW03] Elias Pampalk, Werner Goebl, and Gerhard Widmer, *Visualizing Changes in the Structure of Data for Exploratory Feature Selection*, Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (Washington DC), 2003.

[PHH04] Elias Pampalk, Peter Hlavac, and Perfecto Herrera, *Hierarchical Organization and Visualization of Drum Sample Libraries*, Proceedings of the 7th International Conference on Digital Audio Effects (Naples, Italy), 2004, pp. 378–383.

[Poh05] Tim Pohle, *Extraction of Audio Descriptors and their Evaluation in Music Classification Tasks*, Master's thesis, Technische Universität Kaiserslautern, Austrian Research Institute for Artificial Intelligence (ÖFAI), Deutsches Forschungszentrum für Künstliche Intelligenz (DFKI), 2005.

[PPW05a] Elias Pampalk, Tim Pohle, and Gerhard Widmer, *Dynamic Playlist Generation Based on Skipping Behaviour*, Proceedings of the ISMIR International Conference on Music Information Retrieval (London), 2005, pp. 634–637.

[PPW05b] Tim Pohle, Elias Pampalk, and Gerhard Widmer, *Evaluation of Frequently Used Audio Features for Classification of Music into Perceptual Categories*, Proceedings of the Fourth International Workshop on Content-Based Multimedia Indexing (CBMI'05) (Riga, Latvia), 2005.

[PPW05c] ______, *Generating Similarity-Based Playlists Using Traveling Salesman Algorithms*, Proceedings of the International Conference on Digital Audio Effects (Madrid, Spain), 2005, pp. 220–225.

[PRC00] François Pachet, Pierre Roy, and Daniel Cazaly, *A Combinatorial Approach to Content-Based Music Selection*, IEEE Multimedia **7** (2000), no. 1, 44–51.

[PRM02a] Elias Pampalk, Andreas Rauber, and Dieter Merkl, *Content-Based Organization and Visualization of Music Archives*, Proceedings of the ACM Multimedia (Juan les Pins, France), 2002, pp. 570–579.

[PRM02b] ______, *Using Smoothed Data Histograms for Cluster Visualization in Self-Organizing Maps*, Proceedings of the International Conference on Artifical Neural Networks (ICANN'02) (Madrid, Spain), 2002, pp. 871–876.

[PTVF92] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery, *Numerical Recipes in C: The Art of Scientific Computing*, Cambridge University Press, UK, 1992.

[Pvd05] Steffen Pauws and Sander van deWijdeven, *User Evaluation of a New Interactive Playlist Generation Concept*, Proceedings of the ISMIR International Conference on Music Information Retrieval (London), 2005, pp. 638–643.

[PWC04] Elias Pampalk, Gerhard Widmer, and Alvin Chan, *A New Approach to Hierarchical Clustering and Structuring of Data with Self-Organizing Maps*, Journal of Intelligent Data Analysis **8** (2004), no. 2, 131–149.

[PWL01] François Pachet, Gert Westerman, and Damien Laigre, *Musical Data Mining for Electronic Music Distribution*, Proceedings of the WedelMusic Conference, 2001.

[RAPB05] Pierre Roy, Jean-Julien Aucouturier, François Pachet, and Anthony Beurivé, *Exploiting the Tradeoff Between Precision and Cpu-Time to Speed Up Nearest Neighbor Search*, Proceedings of the ISMIR International Conference on Music Information Retrieval (London), 2005, pp. 230–237.

[Rau99] Andreas Rauber, *LabelSOM: On the Labeling of Self-Organizing Maps*, Proceedings of the International Joint Conference on Neural Networks, (Washington, DC), 1999.

[RJ93] Lawrence R. Rabiner and Biing-Hwang Juang, *Fundamentals of Speech Recognition*, Prentice, Englewood Cliffs, NJ, 1993.

[RPM02] Andreas Rauber, Elias Pampalk, and Dieter Merkl, *Using Psycho-Acoustic Models and Self-Organizing Maps to Create a Hierarchical Structuring of Music by Sound Similarities*, Proceedings of the ISMIR International Conference on Music Information Retrieval (Paris, France), 2002, pp. 71–80.

[RTG00] Yossi Rubner, Carlo Tomasi, and Leonidas J. Guibas, *The earth movers distance as a metric for image retrieval*, International Journal of Computer Vision **40** (2000), no. 2, 99–121.

[Sam69] John W. Sammon, *A Nonlinear Mapping for Data Structure Analysis*, IEEE Transactions on Computers **18** (1969), 401–409.

[SB88] Gerard Salton and Chris Buckley, *Term-weighting approaches in automatic text retrieval*, Information Processing and Management **24** (1988), no. 5, 513–523.

[Sch03] Markus Schedl, *An Explorative, Hierarchical User Interface to Structured Music Repositories*, Master's thesis, Vienna University of Technology, Institut für Medizinische Kybernetik und Artificial Intelligence der Universität Wien, 2003.

[SK05] Richard Stenzel and Thomas Kamps, *Improving Content-based Similarity Measures by Training a Collaborative Model*, Proceedings of the ISMIR International Conference on Music Information Retrieval (London), 2005, pp. 264–271.

[Sku04] André Skupin, *The World of Geography: Mapping a Knowledge Domain with Cartographic Means*, Proceedings of the National Academy of Sciences **101** (2004), no. 1, 5274–5278.

[SKW05a] Markus Schedl, Peter Knees, and Gerhard Widmer, *Discovering and Visualizing Prototypical Artists by Web-Based Co-Occurrence Analysis*, Proceedings of the ISMIR International Conference on Music Information Retrieval (London), 2005, pp. 21–28.

[SKW05b] ———, *A Web-Based Approach to Assessing Artist Similarity Using Co-occurrences*, Proceedings of the Workshop on Content-Based Multimedia Indexing, 2005.

[SP01] Christian Spevak and Richard Polfreman, *Sound Spotting – A Frame-Based Approach*, Proceedings of the ISMIR International Symposium of Music Information Retrieval (Bloomington, IN), 2001, pp. 35–36.

[SVN37] Stanley Smith Stevens, John Volkman, and Edwin B. Newman, *A scale for the measurement of the psychological magnitude of pitch*, Journal of the Acoustical Society of America **8** (1937), 185–190.

[SZ05] Nicolas Scaringella and Giorgio Zoia, *On the Modeling of Time Information for Automatic Genre Recognition Systems in Audio Signals*, Proceedings of the ISMIR International Conference on Music Information Retrieval (London), 2005, pp. 666–671.

[TC02] George Tzanetakis and Perry Cook, *Musical Genre Classification of Audio Signals*, IEEE Transactions on Speech and Audio Processing **10** (2002), no. 5, 293–302.

[Ter68] Ernst Terhardt, *Über Akustische Rauhigkeit und Schwankungsstärke (On the Acoustic Roughness and Fluctuation Strength)*, Acustica **20** (1968), 215–224.

[THA04] Mark Torrens, Patrick Hertzog, and Josep-Lluís Arcos, *Visualizing and Exploring Personal Music Libraries*, Proceedings of the ISMIR International Conference on Music Information Retrieval (Barcelona, Spain), 2004, pp. 421–424.

[TWV05] Rainer Typke, Frans Wiering, and Remco C. Veltkamp, *A Survey of Music Information Retrieval Systems*, Proceedings of the ISMIR International Conference on Music Information Retrieval (London), 2005, pp. 153–160.

[US90] Alfred Ultsch and H. Peter Siemon, *Kohonen's Self-Organizing Feature Maps for Exploratory Data Analysis*, Proceedings of the International Neural Network Conference (INNC'90) (Dordrecht, Netherlands), Kluwer, 1990, pp. 305–308.

[vGV05] Rob van Gulik and Fabio Vignoli, *Visual Playlist Generation on the Artist Map*, Proceedings of the ISMIR International Conference on Music Information Retrieval (London), 2005, pp. 520–523.

[vGVvdW04] Rob van Gulik, Fabio Vignoli, and Huub van de Wetering, *Mapping Music In The Palm Of Your Hand, Explore And Discover Your Collection*, Proceedings of the ISMIR International Conference on Music Information Retrieval (Barcelona, Spain), 2004, pp. 409–414.

[VP05] Fabio Vignoli and Steffen Pauws, *A Music Retrieval System Based on User-Driven Similarity and its Evaluation*, Proceedings of the ISMIR International Conference on Music Information Retrieval (London), 2005, pp. 272–279.

[WC04] Kristopher West and Stephen Cox, *Features and classifiers for the automatic classification of musical audio signals*, Proceedings of the ISMIR International Conference on Music Information Retrieval (Barcelona, Spain), 2004, pp. 531–536.

[WC05] ______, *Finding An Optimal Segmentation for Audio Genre Classification*, Proceedings of the ISMIR International Conference on Music Information Retrieval (London), 2005, pp. 680–685.

[WE04] Brian Whitman and Dan Ellis, *Automatic Record Reviews*, Proceedings of the ISMIR International Conference on Music Information Retrieval (Barcelona, Spain), 2004, pp. 470–477.

[WL02] Brian Whitman and Steve Lawrence, *Inferring Descriptions and Similarity for Music from Community Metadata*, Proceedings of the 2002 International Computer Music Conference (Göteborg, Sweden), 2002, pp. 591–598.

[WS02] Brian Whitman and Paris Smaragdis, *Combining Musical and Cultural Features for Intelligent Style Detection*, Proceedings of the 3rd International Conference on Music Information Retrieval (Paris, France), 2002, pp. 47–52.

[YP97] Yiming Yang and Jan O. Pedersen, *A comparative study on feature selection in text categorization*, Proceedings of ICML-97, 14th International Conference on Machine Learning (Nashville, US) (D. H. Fisher, ed.), Morgan Kaufman Publishers, San Francisco, US, 1997, pp. 412–420.

[ZF99] Eberhard Zwicker and Hugo Fastl, *Psychoacoustics, Facts and Models*, 2nd ed., Springer, Berlin, 1999.

[ZF04] Mark Zadel and Ichiro Fujinaga, *Web Services for Music Information Retrieval*, Proceedings of the ISMIR International Conference on Music Information Retrieval (Barcelona, Spain), 2004, pp. 478–483.

[ZP04] Aymeric Zils and François Pachet, *Automatic Extraction Of Music Descriptors From Acoustic Signals*, Proceedings of the ISMIR International Conference on Music Information Retrieval (Barcelona, Spain), 2004, pp. 353–356.

Elias Pampalk

I was born in 1978 in Maputo (Mozambique). From 1996 to 2006 I studied computer science at the Vienna University of Technology. In 2001 I wrote my Master's thesis on "Islands of Music: Analysis, Organization, and Visualization of Music Archives" [Pam01]. In 2006 I completed my studies with the defense of this doctoral thesis.
From 2002 to 2006 I worked at the Austrian Research Institute for Artificial Intelligence (OFAI) under the supervision of Gerhard Widmer. From 2002 to 2003 I contributed to the project "Computer-Based Music Research: Artificial Intelligence Models of Musical Expression". From 2004 to 2006 I contributed to the European project SIMAC (Semantic Interaction with Music Audio Contents).

The picture on the right shows the making of this thesis. Those two machines did most of the work and most of the figures and tables on the wall did not make it into the final version (they were part of the experiments described in Chapter 2). The picture was taken at my OFAI desk on January 6, 2006.