DIPLOMARBEIT

Tool Support for Acoustic Evaluation of Music Similarity

ausgeführt am Institut für
Softwaretechnik und Interaktive Systeme
der Technischen Universität Wien

unter der Anleitung von
ao.univ.Prof. Dr.techn. Andreas Rauber
und Dipl.Ing. Thomas Lidy

durch

ANDREAS SENFTER

Mandlgasse 1/35
1120 Wien

Wien, am 15. März 2008

Acknowledgements

I would like to acknowledge the support of Andreas Rauber and Thomas Lidy in my thesis.

In particular I would like to thank my family for their unconditional emotional and financial support during my study.
I am extremely grateful to Benjamin who always gave me strength and motivation for my study.
I wish to express my gratitude to Markus and Bernhard for their patience, endless discussions and helpful considerations for my thesis.
I would like to thank all my friends who were involved in the user study for taking the time for the listening test.

Abstract

The increasing number of music libraries offering an ever growing number of tracks on public and private domains stimulated the demand for innovative approaches to let users access their music. Fast and efficient access to requested audio data in huge music collections requires intuitive organization structures. Various different structuring methods have been applied to this problem, many based on the specification of a similarity measurement between the music tracks. This kind of organization process represents the human, intuitive procedure of structuring. Technical systems implementing this structuring approach, namely music similarity systems, characterize elements of big music repositories based on configurations of features extracted from each song of the collection. Such a feature vector describes the underlying audio content by capturing important aspects of music, e.g. pitch, timbre, tempo, beat. Using these vectors, several innovative techniques, e.g. the Self-Organizing Map, perform an organization of the whole music collection. In order to offer a graphical representation to the user, topology-preserving mapping techniques perform a projection of those multidimensional descriptors to a 2-dimensional lattice. This enables the user to explore the music by moving across a plane.

The evaluation of the quality of similarity measures between pieces of music is a non-trivial, sophisticated task because human cognition of music and perception of similarity inherently is biased by subjective interpretation and reasoning which usually is based on knowledge and conventions of the real world. Even songs belonging to the same genre, having a similar structure might be recognized as completely dissimilar by human cognition. Therefore, the evaluation of similarity between music tracks typically requires human judgment.

This work focuses on the acoustic evaluation of music similarity. It presents a novel tool called cLynx which offers several different features for an efficient acoustic evaluation of music similarity and additionally enables exploring and analyzing structured audio repositories much faster and more efficiently. An evaluation based on a listening test studies the applicability of cLynx in various listening situations by highlighting trends in the subjects' perceptions and analyses connections between subjective interpretation and individual preferences and the subjects' perception of music similarity and susceptibility to outlier detection.

Kurzzusammenfassung

Die starke Zunahme von öffentlichen und privaten Musikarchiven, zusammen mit dem stetigen Wachstum an angebotenen Musikstücken, forderte die Entwicklung innovativer Zugangsmöglichkeiten zur Musik. Ein schnelles und effizientes Auffinden gewünschter Musiktitel in riesigen Musiksammlungen erfordert eine leicht verständliche, intuitive Organisation der zugrunde liegenden Daten. Viele solcher Strukturierungsansätze basieren auf der Definition eines Ähnlichkeitsmaßes zwischen den einzelnen Musikstücken, was im Grunde der alltäglichen menschlichen Strukturierungsgewohnheit entspricht. Die Grundlage der technischen Umsetzung bildet die Beschreibung einzelner Werke anhand von mehreren Merkmalen, die aus den Stücken extrahiert werden. Die Gesamtheit der gewonnenen Merkmale bildet einen so genannten Feature Vector, welcher die Struktur des Musikstückes beschreibt. Unter Verwendung dieser Vektoren vollbringen innovative Techniken aus dem Gebiet des maschinellen Lernens (z.B. Self Organizing Map) die Strukturierung eines gesamten Musikarchivs.

Die subjektive Auffassung und Bewertung von Ähnlichkeiten zwischen musikalischen Werken sind von Mensch zu Mensch verschieden, da die musikalische Wahrnehmung und das Ähnlichkeitsempfinden in natürlicher Weise vom Prozess der subjektiven Interpretation und der logischen Schlussfolgerung abhängig sind. Somit ist es möglich, dass eine Musiksammlung Stücke derselben Struktur beinhaltet aber von der menschlichen Wahrnehmung als absolut unähnlich aufgefasst wird. Aus diesem Grund ist die Evaluierung von Ähnlichkeitsmaßen zwischen Musikstücken eine hoch komplizierte Aufgabe, die hauptsächlich mit Hilfe von Hörstudien durchgeführt wird.

Diese Diplomarbeit richtet ihr Hauptaugenmerk auf die akustische Evaluierung von Ähnlichkeitsmaßen. Dazu wird das Tool cLynx präsentiert, welches mehrere unterschiedliche Funktionen zur Unterstützung akustischer Evaluierung bereitstellt und des Weiteren ein effizienteres Erkunden von strukturierten Musikarchiven ermöglicht. Das Tool cLynx wurde in einer Hörstudie verwendet, um seine Eignung in unterschiedlichen Hörsituationen zu analysieren und um Beziehungen zwischen der subjektiven Interpretation und der individuellen Vorliebe von Testobjekten und deren musikalischen Ähnlichkeitsempfinden und Empfindlichkeitsmaß zur Erkennung unpassender Passagen zu erkunden.

Contents

List of Tables **xii**

List of Figures **xiii**

1 Introduction **1**

1.1 Motivation 1

1.2 Outline 3

1.3 Contributions 4

2 Related Work **5**

2.1 Unsupervised Learning 6

2.1.1 Clustering 6

2.1.2 Dimensionality Reduction 10

2.1.3 Artificial Neural Networks and Self-Organizing Maps 13

2.2 PlaySOM 14

2.2.1 Interface and Interaction 15

2.2.2 Sky-Metaphor Visualization 16

2.3 Control Amplifiers 17

2.3.1 Dynamic 19

2.3.2 Compressor 21

2.3.3 Limiter 24

2.3.4 Expander 24

2.3.5 Gates 24

2.4 Evaluation of Acoustic Similarity 24

2.4.1 MIREX 25

2.4.2 Subjective Evaluations 26

2.5 Summary 27

3 cLynx **29**

3.1 Overview 30

3.2 Components 30

3.2.1 Analysis Component 32
3.2.2 Normalization Component 34
3.2.3 Mixer Component 38
3.2.4 Extending cLynx 50
3.3 Run-Time Modes 51
3.3.1 Stand-Alone Mode 52
3.3.2 Console Mode 53
3.3.3 GUI Mode 53
3.3.4 Plug-In Mode 54
3.4 Summary 55

4 Evaluation Results 57
4.1 Test-Set 58
4.2 Analysis Tuning 58
4.2.1 Procedure 59
4.2.2 Random Selection Approach 60
4.2.3 Uniform Selection Approach 63
4.2.4 Summary 64
4.3 User Study 64
4.3.1 Parameter Analysis 64
4.3.2 Subjective Evaluation 68
4.4 Summary 78

5 Conclusions and Future Work 79

A Parameter Files 81

B User Study 83

Bibliography 98

List of Tables

2.1 Estimated guide values of the dynamic range of the human voice, several classes of instruments and orchestras. 20
2.2 Dynamic range of several studio devices. 22

3.1 All keys of the setting file which passes all required parameters to cLynx in the Stand-Alone Console Mode. 52

4.1 Organization of the ISMIRgenre test set 58
4.2 Averaged, normalized differences of Δ_{50} and a π of 10 for the genres Classic and Metal. 60
4.3 Several configurations of the Dynamic Sequential Mixer. 65
4.4 Correct genre identification and difference of the detected number of songs to the real number of songs 66
4.5 Detailed overview of all listening situations. 70

A.1 Possible keys and values of the start-parameter file. 81

B.1 Questionnaire of the pre-study. 83
B.2 Questions each subject was asked after every listening situation. 84
B.3 Questions each subject was asked after finishing all listening situations. . . 84
B.4 Results of Subject1 85
B.5 Results of Subject2 86
B.6 Results of Subject3 87
B.7 Results of Subject4 88
B.8 Results of Subject5 89
B.9 Results of Subject6 90
B.10 Results of Subject7 91
B.11 Results of Subject8 92
B.12 Results of Subject9 93
B.13 Results of Subject10 94
B.14 Results of Subject11 95

B.15 Results of Subject12 96
B.16 Results of Subject13 97

List of Figures

2.1 Classification of clustering approaches [Jain et al., 1999] 8
2.2 Graphical Interface of the PlaySOM application. 15
2.3 Detailed views of the PlaySOM. 15
2.4 Sky Visualization and several displacement scenarios. [Latif and Mayer, 2007] 17
2.5 Static characteristic of the basic Control Amplifier types [Raffaseder, 2002]. 18
2.6 The human's audibility limit in decibel in dependency of the frequency [Dickreiter, 1997]. 20
2.7 Several compression ratios and their effect to the output signal. 23
2.8 Evalutron 6000 GUI components (source: MIREX web page). 26

3.1 cLynx's itinerary 31
3.2 Linear mapping of a Linear Amplitude Normalization 36
3.3 The Dynamic Sequential Mixer plays back one track per channel. 41
3.4 Two and four channel mode of the Distance Mixer. 47
3.5 cLynx's graphical user interface. 53

4.1 Trend of δ_{max}, δ_{min} and δ_{σ} for the sets Δ_{70} to Δ_{10} and a π of 10 for the genre Classic. 61
4.2 Several differences of various Δ sets for varying partitioning. 62
4.3 $\delta_{max}, \delta_{min}$ and δ_{σ} for various genres on uniformly distributed sample takes . 63
4.4 Participant's feeling of duration and identification of DPM's highlighting effect. 67
4.5 Genre based segmentation of all listening situations. 69
4.6 Every region of all listening situations marked in the music map. 71
4.7 Voice detection and lyrics understanding according to all mixer types. . . . 72
4.8 Trends of the estimation of the number of perceived songs. 73
4.9 Ability of genre detection related to the mixer type. 74
4.10 Several trends from parallel to sequential playback. 75
4.11 Subjects' perception of the different mixer types. 76
4.12 Results of the over-all questionnaire presented in table B.3. 77

Chapter 1

Introduction

1.1 Motivation 1
1.2 Outline 3
1.3 Contributions 4

1.1 Motivation

The World Wide Web represents a medium offering a huge amount of digital data, e.g. text, audio, video, pictures. Especially audio massively gained in importance over the last few years. Beginning with Napster, Morpheus and Co digital music got a still active boost of popularity and received a mass appeal quasi over night. More and more steadily growing online audio archives arise in order to accomplish the growing demand of digital audio. In parallel to the increasing library sizes, the problem of giving the recipient a general review of the range of products in a fast and simple way raised. Therefore, there has been tireless effort in science to develop innovative organization techniques which aim at presenting a clear, intuitive arrangement to the users. A simple but time-expensive structuring approach, which was applied to this problem, used manual assignment of every single song to some kind of meta-information. E.g., formats like MPEG-1 Audio Layer 3 (more commonly referred to as MP3) store meta-information specifying the work's basic data. But this meta-data whether assures correctness, completeness and availability. Therefore, science was challenged in developing structuring approaches which do not rely on such kind of pre-processing task. So science directed its focus on the development of novel organization techniques which are based on the musical content of audio data.

Music Information Retrieval (MIR) research deals with content based analysis of music. Within this research field several approaches for automatic detection of semantic features were developed which cover musical information like loudness, tempo, beat, rhythm, timbre, pitch, harmonics, melody etc. The retrieval of features tackles the issue from two different angles. While one approach deals with directly extracting a set of features from an audio signal, another technique generates features based on transformations or combinations of the extracted features. The specification of an adequate configuration of retrieved features (called feature vector) which describes a music signal best is a very sophisticated process and inter-alia depends on issues of the defined task and psycho-acoustic matters. The feature vector of a piece of music builds the basis for:

- retrieving/searching songs based on sample music
- automatic music classification
- music organization by similarity
- music identification

The songs' feature vectors allow the definition of acoustic distance and musical similarity measures between the tracks to all others. Several similarity based structuring approaches have been applied in MIR research, e.g. to music organization by similarity. Many of these methods are based on clustering techniques (e.g. Self-Organizing Map) because of their ability of representing the fuzziness and overlapping of different genres. Furthermore, topology-preserving mapping techniques were applied to project the song's multidimensional feature vector to a 2-dimensional lattice in a way that adjoining tracks are more similar according to the defined similarity measure than songs which are a long way away from each other. Several graphical description approaches have been developed which visualize a 2-dimensional representation of music collections. A very intuitive visualization technique, the so called Music Map, resembles a geographic map in order to generate virtual landscapes formed by a music archive. In the course of this thesis a tool called PlaySOM will be reviewed which consists of a big repertoire of different visualization methods and enables an interactive exploration of music collections.

The human cognition of music and perception of similarity inherently is biased by subjective interpretation and reasoning which usually is based on knowledge and conventions of the real world. Even songs belonging to the same genre, having a similar structure might be recognized as completely dissimilar by human cognition. Therefore, the evaluation of the performance of music similarity measures between pieces of music is a non-trivial, sophisticated task and is challenged by the differing human cognition of music and the subjective interpretation of whether a track is similar or dissimilar to a query song. As a consequence, the evaluation of the similarity between music tracks typically requires human judgment. In recent years various evaluation contests have been arranged within several International Conferences on Music Information Retrieval (ISMIR[1]) since ISMIR

[1]ISMIR is derived from International Symposium on Music Information Retrieval which was redefined to International Conference on Music Information Retrieval between ISMIR 2001 and ISMIR 2002

2004. During the 6^{th} ISMIR Conference in 2005 the 1^{st} separated evaluation project, called Music Information Retrieval Evaluation eXchange (MIREX), was conducted. Large scale music similarity evaluation, which primarily relays on human judgment, requires systems which assist the subjects during the evaluation process. For this purpose, Stephen J. Downie's IMIRSEL lab contributed the so called Evalutron 2000 (E2K) to the subjective evaluation of the MIREX 2006 which supported human judges in evaluating algorithms submitted to the 'Audio Music Similarity and Retrieval' and 'Symbolic Melodic Similarity' tasks. While this evaluation system forced to listen to the songs consecutively, the cLynx evaluation tool presented in this thesis also enables simultaneous playback, or sequential playback with different emphasis.

1.2 Outline

This thesis is structured into three main chapters:

Chapter 2 introduces into state-of-the-art approaches of areas relevant for this thesis. Methods of unsupervised learning such as Clustering and Dimensionality Reduction are discussed and an Artificial Neural Network, called Self-Organizing Map (SOM) is presented. Thereafter, a SOM-based application, namely PlaySOM is introduced and the 'Sky-Metaphor Visualization' –a novel visualization method– is explained. Next, the basics of signal processing relevant for this work are presented. In the last part of this chapter the related work of human evaluations of audio-based similarity measures are briefly reviewed.

Chapter 3 presents the cLynx application which was implemented in the course of this thesis. All components are discussed in detail including a guide for further extensions and an explanation of cLynx's various execution modes. Additionally, developers of MIR-systems are guided in how to integrate the tool into their system.

In Chapter 4 two evaluations are published. The first experiment introduces an approach of tuning the cLynx's analysis process by approximating the original analysis results on the basis of only a few song parts. In the second part a listening test is presented which is based on the PlaySOM application using the cLynx plug-in. The evaluation is focused on the performance of the plug-in in supporting the user in analyzing and exploring music data and in evaluating an underlying similarity measure. Furthermore, it highlights trends of human cognition and similarity interpretation.

Chapter 5 provides several suggestions for possible improvements and useful extensions of the cLynx tool and draws conclusions.

1.3 Contributions

The contributions of this work can be divided into three categories:

- Chapter 3 deals with the development of a novel tool, namely cLynx, for acoustic validation of similarity between pieces of music. It additionally offers features for analyzing, browsing and exploring music data of structured audio collections.
- Chapter 4 presents a listening test which studies the human cognition of music and subjective interpretation of music similarity using cLynx. It analyses the various features offered by the cLynx application according to its performance, effectiveness and expressiveness.
- The cLynx application is presented using input data passed by an underlying SOM based Music Similarity System called PlaySOM by extending the application as plug-in.

Chapter 2

Related Work

2.1 Unsupervised Learning 6

2.1.1 Clustering 6

2.1.2 Dimensionality Reduction 10

2.1.3 Artificial Neural Networks and Self-Organizing Maps 13

2.2 PlaySOM 14

2.2.1 Interface and Interaction 15

2.2.2 Sky-Metaphor Visualization 16

2.3 Control Amplifiers 17

2.3.1 Dynamic 19

2.3.2 Compressor 21

2.3.3 Limiter 24

2.3.4 Expander 24

2.3.5 Gates 24

2.4 Evaluation of Acoustic Similarity 24

2.4.1 MIREX 25

2.4.2 Subjective Evaluations 26

2.5 Summary 27

This chapter briefly introduces into the unsupervised machine learning approaches Clustering and Dimensionality Reduction followed by the presentation of an Artificial Neural Network, called Self-Organizing Map. Thereafter, the PlaySOM is discussed which is an alternative method to song selection and play-list generation in large music archives based on a Self-Organizing Map approach followed by an introduction into signal processing where an insight into Control Amplifiers (Compressor, Limiter, Expander and Noise Gates) is given. The last section of this chapter reviews related work of human evaluations of audio-based similarity measures.

2.1 Unsupervised Learning

The field of machine learning is concerned with the question of how to construct algorithms that automatically improve with experience [Mitchel, 1997], which borrow and build up ideas from several fields, e.g. statistics, computer science, cognitive science, mathematics, etc. In the sub-field of machine learning, known as unsupervised learning the machine receives some sequence of input data $x_1, x_2, x_3, ...$ where x_t defines the input at time t. Unlike supervised learning where the model additionally obtains a sequence of desired outputs $y_1, y_2, y_3, ...$, the system receives no information about the output. Despite the absence of supervised target outputs or feedback from the environment the system's goal is to find a useful representation of the underlying data which can be thought of finding patterns which can be used for:

- Outlier Detection
- Data Compression

Unsupervised learning algorithms explore the input data in order to discover an inner structure, e.g. finding clusters within the data. A cluster covers data items which are similar in some kind of features, but the clusters itself are disparate from each other. The learning can be seen as learning an estimated model which represents the probability distribution of a new input x_t resulting from all previous inputs $x_1, x_2, ..., x_{t-1}$ defined as [Ghahramani, 2004]:

$$P(x_t|x_1, ..., x_{t-1}) \tag{2.1}$$

In the following paragraphs the two most common examples of unsupervised learning, Clustering and Dimensionality Reduction, are discussed in more detail.

2.1.1 Clustering

Clustering algorithms are methods to divide a set of n observations into g groups so that members of the same group are more alike than members of different groups [Ripley, 1996]. These groups are called clusters. The clustering algorithm can be seen as a mapping that assigns each observation to a cluster. Some algorithms are closely linked with Vector Quantization [Gersho and Gray, 1992] which is defined as the mapping Q of an observation, a vector (input vector) in k-dimensional space $\mathbb{R}^k$, into a finite set C:

$$Q: \quad \mathbb{R}^k \rightarrow C \tag{2.2}$$

C represents the so called codebook which contains distinct elements $\{y_1, y_2, \ldots, y_n\}$, $y_i \in \mathbb{R}^k$ for $1 \leq i \leq n$. Each element of the codebook is a vector representing a cluster center.

So every input vector x_i can be assigned with a cluster center y_i. In order to assign similar input data with the same cluster center a similarity and distance measure has to be defined. Both measures build the basis for the quality of the clustering process. The similarity of two objects o_i and o_j of the set O can be expressed by the function σ defined as:

$$\sigma : O \times O \quad \rightarrow \quad \sigma(o_i, o_j) \in [0, 1] \tag{2.3}$$

which performs a mapping into the defined output space of [0,1] where 0 defines low similarity and 1 equal. The resulting matrix, called the similarity matrix, is a symmetric matrix with 1s down in its main diagonal because every element is equal to itself. The distance between two elements o_i and o_j of the set O can be expressed by the function δ defined as:

$$\delta : O \times O \quad \rightarrow \quad \delta(o_i, o_j) \in \mathbb{R}^+ \tag{2.4}$$

As a measure for the distance between the objects o_i and o_j several metrics have been proposed. The probably most common is the L2-Norm, also referred to as Euclidean distance:

$$\delta(o_i, o_j) = \sqrt{\sum_{k=0}^{n} (o_{i,k} - o_{j,k})^2} \qquad o_i, o_j \in \mathbb{R}^n \tag{2.5}$$

where $o_{i,k}$ represents the k^{th} index of the object o_i. Another commonly used distance metrics is the L1-Norm which is also called City-block or Manhattan distance:

$$\delta(o_i, o_j) = \sum_{k=0}^{n} |o_{i,l} - o_{j,l}| \qquad o_i, o_j \in \mathbb{R}^n \tag{2.6}$$

The smaller the distance the higher is the degree of similarity. The generated symmetric matrix is called distance matrix and has 0s down in its main diagonal because the distance from each element to itself is 0. Furthermore, there is a relation between the similarity and distance which allows extraction from each other by transformations. [Bock, 1974] discusses three common transformations of how to get the distance function δ of the similarity function σ. The more complex transformation in the opposite direction is explained by [Steinhausen and Langer, 1977].

There are several methods to clustering data. [Jain et al., 1999] organize the various approaches based on [Jain and Dubes, 1988] as depicted in figure 2.1. They distinguish between hierarchical and partitioned approaches in the top level which are discussed in detail

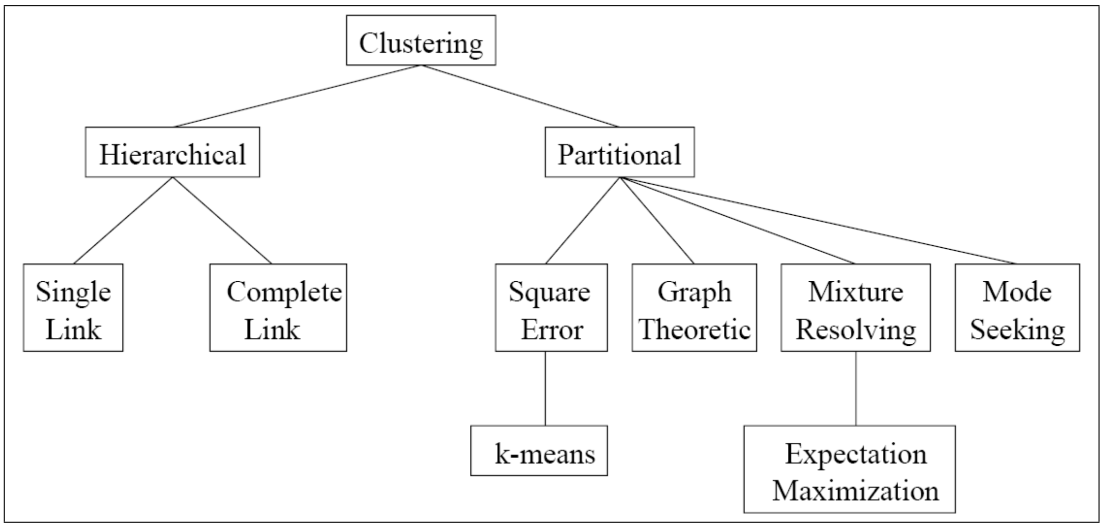


Figure 2.1: Classification of clustering approaches [Jain et al., 1999]

in the next paragraphs. A broader classification of clustering algorithms in the top level is done by [Berkhin, 2002]. Next to hierarchical and partitioned approaches, Berkhin additionally differs between grid-based methods, constraint-based clustering, methods based on co-occurrence of categorical data, clustering algorithms used in machine learning, scalable clustering algorithms and algorithms for high dimensional data.

Partitional Clustering

A partitional clustering algorithm divides the data into a pre-assigned number of groups which is accompanied by the problem of the choice of the number of desired output clusters. [Dubes, 1987] provides instructions on this key design decision. Besides the number of desired clusters, an optimum criterion (e.g. variance criterion [Bock, 1974]) has to be defined. Usually these algorithms create clusters by fulfilling the criterion either locally or globally. The optimum's search within the set of all possible clustering assignments is clearly computationally prohibitive and therefore the algorithm is executed multiple times with different starting states. The best resulting configuration over all runs is used as the output cluster. E.g., if $\{o_1, o_2, \ldots, o_n\}$ are input data, the partitional clustering algorithm C tries to assign each element to k clusters $\{g_1, g_2, \ldots, g_k\}$ in a way so that one element relates to one cluster and one cluster can cover several elements defined as:

$$C: \quad = \{o_1, o_2, ..., o_n\} \rightarrow \{g_1, g_2, ..., g_k\} \tag{2.7}$$

The most common and frequently used partitional clustering techniques use the squared error criterion. The squared error for a partition A which contains k clusters is defined as:

$$q(A) = \sum_{i=1}^{k} \sum_{x_j \in A_i} |x_j - \widehat{x}_{A_i}|^2 \tag{2.8}$$

where x_j is the j^{th} observation belonging to the i^{th} cluster A_i and $\widehat{x}_{A_i}$ is the center of A_i. The algorithms start with some initial division of the input vectors into a fixed number of clusters or cluster centers. In the next step each input vector is reassigned to its closest cluster and the new cluster centers are recomputed. This process is repeated until a convergence criterion is met (e.g. the cluster membership is stable or the squared error discontinues decreasing significantly).

The most common partitional clustering algorithm which employs a squared error criterion is called k-Means. The algorithm starts by creating an initial partition of the input vectors with k clusters having k randomly-chosen input vectors or defined points inside the hyper volume as their center. Next, every input vector is assigned to its closest cluster and the cluster center is moved into the group's center. It continues reassigning the input vectors to clusters based on the similarity between the input vectors and the cluster center and recomputing the cluster center by using the current cluster membership until a convergence criterion is met. The main problem of k-Mean is the selection of an initial partition. A badly chosen start partition may converge to a local minimum.

Another sub-class of partitional clustering techniques represents algorithms which use graph-theoretic methods for clustering which consider the whole input data as a graph. The similarity between the data usually is expressed by the edges. The most frequently used graph-theoretic approach is based on the idea of the minimum spanning tree (MST) which is considered by [Zahn, 1971]. After the construction of the MST the longest edges are deleted in order to generate clusters. The algorithm is discussed and illustrated by pseudo code in [Brandes et al., 2003].

The mixture-resolving and mode-seeking clustering techniques assume that the input vectors are drawn from one of several distributions. These algorithms try to identify the parameters and the number of these distributions. Usually, it is assumed that the distribution types are known (e.g. Gaussian). Traditional approaches for estimating the parameters are based on maximizing the likelihood of the parameters for the data which are discussed in [Bishop, 1995]. A general-purpose maximum likelihood algorithm called Expectation Maximization (EM) algorithm [Dempster et al., 1977] has been applied to the problem of parameter estimation.

Hierarchical Clustering

Instead of partitional clustering approaches which obtain a single partition, the hierarchical clustering algorithms produce a sequence of partitions. Each partition is nested in the next higher partition in a way so that the partition in the root covers the whole set of all observations and the partitions of the lowest level only contain a single observation. Such hierarchical methods yield a tree of nested groups of observations which can be graphically represented by a dendrogram. The bigger part of the hierarchical algorithms uses variants of the single-linkage or complete-linkage methods which both describe the similarity between a pair of clusters. In the single-linkage method the distance D between two clusters A_r, A_s is the minimal distance d between all assigned observations $k \in A_r$, $j \in A_s$. The clusters A_r and A_s of the partition $A^{\nu-1}$ having the nearest neighboring objects are merged in iteration ν defined as [Bock, 1974]:

$$D_{A_r A_s} := \min_{k \in A_r, j \in A_s} \{d_{kj}\} = \min_{i \neq j} \{D_{A_i, A_j}\} \quad =: D_v \tag{2.9}$$

That means, although having only two sufficient equal objects two clusters are merged. Relating to the graph-theoretic approaches all information required for the single-linkage clustering of a set of points can be derived from the minimum spanning tree [J. Gower, 1969]. In the complete-linkage method the distance between two clusters is the maximum of all distances between the observations in the two clusters. Therefore, the clusters A_r and A_s of the partition $A^{\nu-1}$ having smallest maximum distance are merged in iteration ν.

$$D_{A_r A_s} := \max_{k \in A_r, j \in A_s} \{d_{kj}\} = \min_{i \neq j} \{D_{A_i, A_j}\} \quad = D_\nu \tag{2.10}$$

Another aspect to classify hierarchical clustering methods is the methodic of creating the partition tree. The agglomerative clustering algorithms start by placing each observation in its own cluster and iteratively merge subsets to larger clusters. In contrast to that, a hierarchical divisive algorithm starts with a coarse cluster which contains all observations and continues dividing the clusters until the partition contains clusters of only a single observation.

2.1.2 Dimensionality Reduction

One of the biggest challenges throughout machine learning is the handling of high dimensional datasets (set of input vectors). Traditional methods reach their limits partly because of a high number of observations but mostly because of a huge number of variables representing them. The number of variables (features) associated with the observation is called its dimension. The problem which often occurs with high dimensionality is that not all measured features are as important for representing the observation as others [Fodor, 2002] or include redundancy (features correlate with each other) [Carreira-Perpinan, 1997]. Furthermore, in many cases the underlying data is wanted to be visualized but this is not reasonably possible for more than three dimensions [1]. Additionally, Dimensionality Reduction is a strategy to avoid the 'curse of dimension' problem [Bellman, 1961]. However, there are several motives for reducing the dimension of the underlying data.

As discussed in the previous section 2.1.1 clustering algorithms represent input vectors by assigning them to corresponding clusters. By contrast, dimension reduction techniques decrease the number of features of the input by a mapping ρ to a fewer dimensional representation defined as:

$$\rho : \quad \mathbb{R}^n \rightarrow \mathbb{R}^m \quad (m < n) \tag{2.11}$$

The goal of dimensionality reduction is to reduce the dimensionality of the data while retaining as much as possible of the information of the data. This is possible by stripping out the already mentioned redundant information and thus obtaining a more economic representation of the input. Depending on the domain of the underlying data the dimensionality reduction approaches achieve this reduction by either

- feature selection or
- feature extraction (e.g. transformation and /or combination of feature attributes)

which both are reviewed in [Jain et al., 2000].

The term feature selection refers to algorithms that use the (hopefully) most appropriate subset of the original features in order to represent the data. If therefore X represents the set of all original values and Y is the set all desired values feature selection is defined as

$$[x_1, x_2, ..., x_n] \rightarrow [y_1, y_2, ..., y_m] \tag{2.12}$$

where $x_i, y_i \in X$ and $Y \subset X$. [A. Jain, 1997] presents a taxonomy of feature subset selection algorithms in broad categories and discusses several approaches in detail.

Methods which perform feature extraction are concerned with the creation of new features based on transformations or combinations of the original feature set. The algorithms determine a proper subspace of dimensionality m in the original feature space of dimensionality n with $m < n$. The transformation is accomplished either in a linear or nonlinear way.

Principal Component Analysis

The most common linear feature extraction approach is the Principal Component Analysis (PCA) also referred to Karhunen-Loéve expansion which achieves the reduction by linear

[1]Some approaches for representing data up to five dimensions are reviewed in the first chapter of [Scott, 1992].

transformation to a new set of variables called principal components which are orthogonal linear combinations of the original variables with the largest variance [Scott, 2002]. The new values are ordered in a way that the first few principal components explain most of the variance in the data. So the first value is the linear combination with the largest variance, the second value is orthogonal to the first and represents the second largest variance etc. The number of calculated principal components is the same as the number of original values but the first few retain most of the variance, so that the rest can be discarded by having minimal loss of information.

Multidimensional Scaling

Another linear feature extraction approach is referred to as Multidimensional Scaling (MDS) [I. Borg, 2005] [Cox and Cox, 2000]. A MDS algorithm pairwise analyses the similarity or dissimilarity data between objects, e.g. inter-correlations, similarity ratings, etc. It performs a point to point mapping of the observations of the original multidimensional space $\mathbb{R}^n$ to a low dimensional space $\mathbb{R}^m$ (usually Euclidean) with $m < n$ in such a way that the distances between the points of $\mathbb{R}^m$ match, as well as possible, the original dissimilarity. MDS is motivated by the creation of a graphical representation (2d or 3d) of the underlying data's structure which is easier to understand than an array of numbers by only depicting essential information and smoothing out noise in order to uncover hidden structures. [Kruskal and Wish, 1978] illustrates MDS by an example of use which involves data from an election study. In a survey [H. Weisberg, 1970] each respondent was asked to evaluate his perceptions and preferences of several candidates for the U.S. highest office. In such a case MDS can help locating similarities between the varying evaluations. It supports discerning identifiable features in the public perceptions by mapping the candidates (represented by points in a multidimensional space) to a 2-dimensional representation which provides the hidden structure of the evaluation data (e.g. partnership, ideology).

Suppose the set of points $\{x_1, x_2, \dots, x_k\}$, $x_i \in \mathbb{R}^n$ with the symmetric distance matrix $\Delta = \{\delta_{i,j} \,|1 \leq i, j \leq k\}$. MDS attempts to find a set of points $\{y_1, y_2, \dots, y_k\}$, $y_i \in \mathbb{R}^m$ with the symmetric distance matrix $D = \{d_{i,j} \,|1 \leq i, j \leq k\}$ in a way that the distances between $\delta_{i,j}$ and $d_{i,j}$ are as close as possible according to a corresponding proximity function f:

$$d_{ij} \approx f(\delta_{ij}) \tag{2.13}$$

If f is a continuous parametric monotonic function the MDS approaches are called metric [Cox and Cox, 2000]. If only the rank-order of the proximities is considered the MDS methods are called non-metric.

One of the most common metric scaling approaches is referred to as classical scaling. The method assumes that the dissimilarities are differences and preserves them by coordinates. The approach starts by computing the matrix of distances between all points in e.g. Euclidean space. The eigenvalues λ_i and associated eigenvectors v_i have to be found, where the eigenvectors are normalized so that $v_i^T v_i = \lambda_i$. The eigenvectors build the orthogonal low-dimensional basis vectors for a subspace which represents the highest variance.

2.1.3 Artificial Neural Networks and Self-Organizing Maps

Section 2.1.1 already introduces in several clustering approaches. In this section another extensively used method of clustering is discussed: Artificial neural networks (ANNs). ANNs are inspired by biological ideas and try to imitate the biological neural networks [Zurada, 1992] [Ripley, 1996]. A special subtype of ANNs also referred to as Competitive neural networks [Jain et al., 1996] is often used for clustering data. In contrast to Hebbian learning [Gurney, 1997] where several output units can be activated simultaneously, in competitive learning only one output unit is fired in a way that similar input patterns are grouped and mapped to the same output unit automatically by the network. The calculation of the similarity between the input patterns is based on the correlation in the data.

A well-known and widely used example of ANNs represents the Self-Organizing Map which was introduced by Teuvo Kohonen and therefore also referred to as Kohonen Map [Kohonen, 2001]. SOMs perform a mapping from a high dimensional input space to a lower dimensional output space (usually a 2d lattice) by applying a vector projection in order to reduce the dimensionality and vector quantization (see equation 2.2) in order to find an appropriate representation. The dimensionality reduction is done in a topology preserving way, comparable with Multidimensional Scaling which was discussed in the previous section. Therefore, the SOM usually is utilized to generate a visual representation of a data set in order to ease exploring the structure of the underlying high dimensional data. Similar input data (observations) are represented by a node (also referred to as unit, cell and codebook vector) which all build up the grid the SOM consists of. The units' arrangement on the grid reflects the topology of the data so that neighboring units contain more similar data than the units in the opposing corners.

If a SOM consists of k units and the input vectors are part of the set $\{v_1, v_2, \ldots, v_k\}$, $v_i \in \mathbb{R}^n$, each unit is represented by a so called model vector of the set $\{m_1, m_2, \ldots, m_k\}$, $m_i \in \mathbb{R}^n$. The training process of a SOM stepwise adjusts the map in a way that it more and more optimally describes the domain of observations. The algorithm starts with an initialization of all model vectors m_i (e.g. randomly or with PCA, etc.). During each step of the training phase a randomly selected input vector v_i is assigned to a model vector m_i which is carried by a cell of the lattice. So an input vector is mapped to a unit (best-matching unit) by assigning the input vector to the model vector which is most similar defined as:

$$m_{win} = \min(|v_i - m_i|) \tag{2.14}$$

After each training step all model vectors m_i are adapted by the following step [Ritter and Schulten, 1988] [Kohonen, 1990]:

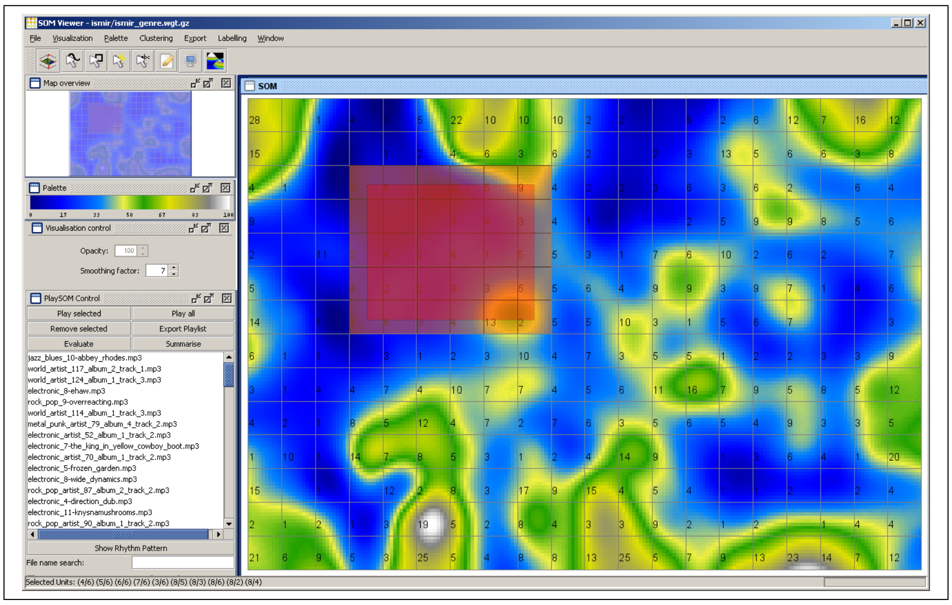
$$m_i^{new} = m_i^{old} + \alpha * h_{ci} * (v_i - m_i^{old}) \tag{2.15}$$

where h_{ci} defines the neighborhood function and α specifies the learning rate which should be a time-decreasing function. The training algorithm of a SOM often ends with a fine-tuning step, where only all best-matching units are adjusted by employing a low learning rate after one more presentation of all input vectors.

Several visualization approaches for Self-Organizing Maps have been developed which are reviewed in [Vesanto, 1999]. The most prominent and widely used method for visualizing the cluster structure of the SOM depicts the distances between the model vectors of neighboring units and is called U-Matrix [Ultsch and Siemon, 1990]. Self-Organizing maps have been applied to a variety of tasks which are listed partly in the SOM Bibliography [Kaski et al., 1998] [Oja et al., 2003] [Pöllä et al., 2006]. Furthermore, the PlaySOM which is presented in the next section browses music collections based on a Self-Organizing Map.

2.2 PlaySOM

The increasing number of music libraries offering an ever growing number of tracks on the World Wide Web stimulated the demand of innovative approaches to let users access their music, e.g. interactive exploration or similarity based search of audio data. The PlaySOM [Neumayer et al., 2005] [Dittenbach et al., 2005] is an interface that enables an interactive exploration of a music collection based on the Self-Organizing Map clustering approach. The underlying map is organized according to the similarity of the audio signals based on the Rhythm Patterns or other feature extraction models [Rauber et al., 2003]. The PlaySOM provides facilities for track selection, play list export and audio playback of selected songs and additionally offers several different visualizations, various approaches of selecting tracks and export to external players. Furthermore, the PlaySOM extends the underlying Self-Organizing Map's operation mode for intuitive exploration, and semantic zooming. That means, it enables the user to zoom into areas of interest and presents the user tracks covered in the selected region which can be selected and played back. My own contribution, the cLynx tool, extends the PlaySOM by features for an acoustic evaluation of the SOM's music clustering abilities which additionally enables quick data exploring.



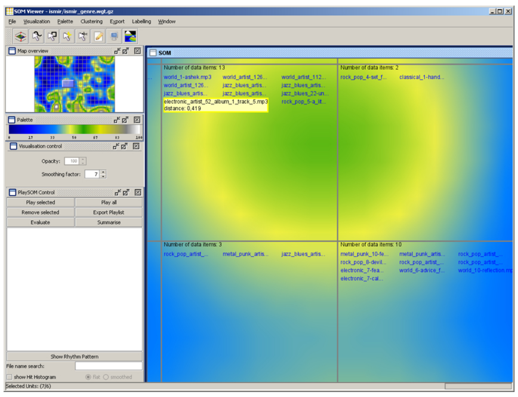
Opacity: 100
Smoothing factor: 7

jazz_blues_10-abbey_rhodes.mp3
world_artist_117_album_2_track_1.mp3
world_artist_124_album_1_track_3.mp3
electronic_8-ehaw.mp3
rock_pop_9-overreacting.mp3
world_artist_114_album_1_track_3.mp3
metal_punk_artist_79_album_4_track_2.mp3
electronic_artist_52_album_1_track_2.mp3
electronic_7-the_king_in_yellow_cowboy_boot.mp3
electronic_artist_70_album_1_track_2.mp3
electronic_5-frozen_garden.mp3
electronic_8-wide_dynamics.mp3
rock_pop_artist_87_album_2_track_2.mp3
electronic_4-direction_dub.mp3
electronic_11-knysnamushrooms.mp3
rock_pop_artist_90_album_1_track_2.mp3

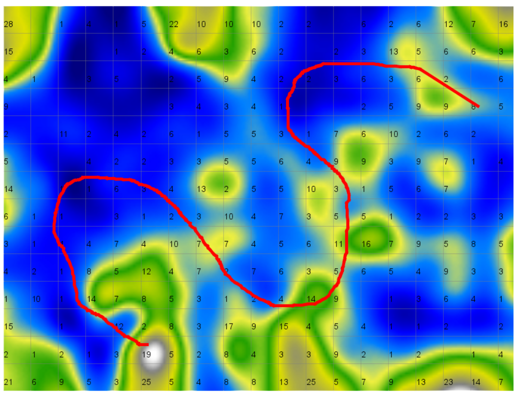
File name search:
Selected Units: (4/6) (5/6) (6/6) (7/6) (3/6) (8/5) (8/3) (8/6) (8/2) (8/4)

Figure 2.2: Graphical Interface of the PlaySOM application.

2.2.1 Interface and Interaction



(a) More detailed labeling with high zoom level.



(b) The path selection model adds tracks of adjoining units to the play list.

Figure 2.3: Detailed views of the PlaySOM.

The PlaySOM's GUI, as shown in figure 2.2 can be divided into two main parts. The left side covers several dialogs presenting an interactive overview of the map, a color palette displaying the color range of the currently active visualization, the visualization control for modifying parameters of the currently used visualization and an interactive play list. The play list interacts with the map and displays all tracks of the map's selected region. On its top, several buttons are located to edit, filter or export the list and to play-back or evaluate the listed tracks. The right side of the application provides the interactive map which can be displayed using various visualizations [Lidy, 2006] and allows exploring the data by moving across the map, zooming into regions of interest and selecting tracks for play-back. The map is partitioned into squares, each representing a SOM-unit and offering a different amount of information depending on the zooming level. Figure 2.2 depicts the map at the outmost zooming factor where the cells are labeled which the number of tracks assigned to the unit. With increasing zooming factor the cells display more detailed information about the songs, as illustrated in figure 2.3(a).

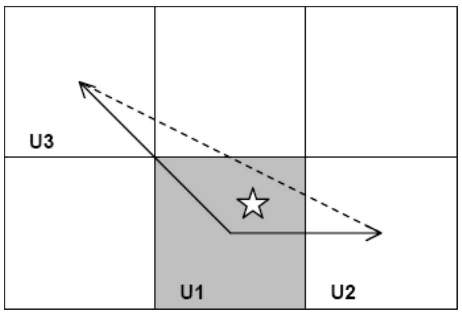
As already mentioned the play list directly interacts with the map enabling the user to generate play lists by applying a selection model supported by the PlaySOM. Depending on the desired genre mixture of tracks in the play list the PlaySOM offers two selection approaches to the user. The rectangular selection method (see figure 2.2) enables user to span a rectangular selection area over several units which results in passing all underlying tracks to the play list. This model enables to generate play lists containing songs from a special (sub-) genre by the selection of the corresponding cluster region. Figure 2.3(b) illustrates the second approach, namely the path selection model, used to build up play lists containing tracks belonging to units which are adjoining a drawn path. By this, the user can generate a play list which contains tracks expressing the change-over from a (sub-) genre to another.

2.2.2 Sky-Metaphor Visualization

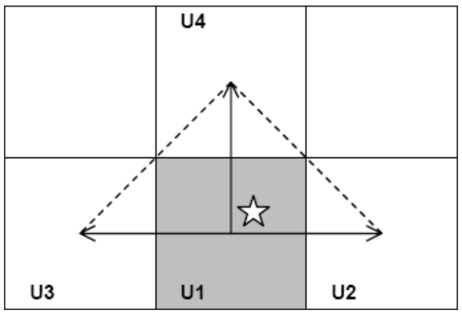
The PlaySOM offers various visualization methods for a graphically illustration of the SOM which can be selected and browsed by the user [Lidy, 2006]. A novel visualization technique added to the PlaySOM represents the so called Sky-Metaphor Visualization [Latif and Mayer, 2007] which aims to improve the understanding of the underlying data by placing each observation within its best-matching unit in respect to the distances of adjoining units. This results in a distributed spatial appearance of the tracks within a unit instead of locating all songs in the center of the corresponding unit and therefore increases the user's understanding of the structure within an unit.

Each song is represented by a lighting star on a black background which should resemble the starry sky. Regions containing many similar objects may form clusters of stars. By contrast, empty units only appear black, resembling a starless area in the night sky. Figure 2.4(c) depicts the Sky-Metaphor Visualization of a trained map.

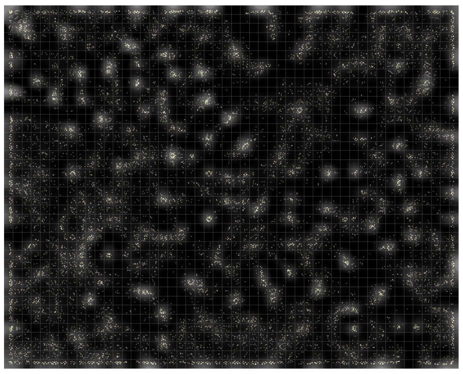
The calculation of a track's exact spatial coordinates within a unit is based on the distance information to the closest units. Metaphorically speaking, the attractive forces of the closest units pull the song in a way that the pull forces of units decrease with increasing distance. This causes a displacement from the center which is insignificant to farther units, as illustrated by figure 2.4(a). $U1, U2, U3$ represent the best, second and third matching units which usually suffice for the calculation of the spatial coordinates defined as



(a) Two neighboring forces on an element.



(b) Three neighboring forces on an element.



(c) Sky Visualization of 20 newsgroup maps.

Figure 2.4: Sky Visualization and several displacement scenarios. [Latif and Mayer, 2007]

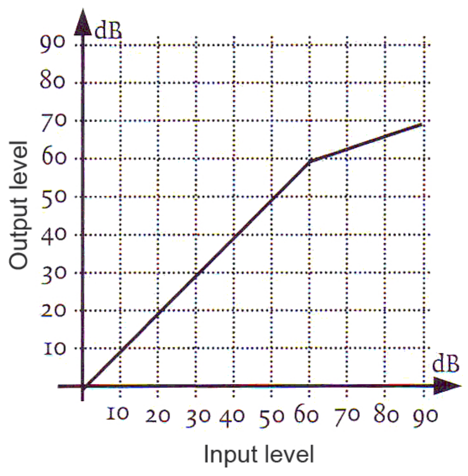
$$P < x, y >= \left\langle \lambda * \sum_{i=2}^{k} F_i * \frac{1}{U_{i<x>} - U_{1<x>}}, \lambda * \sum_{i=2}^{k} F_i * \frac{1}{U_{i<y>} - U_{1<y>}} \right\rangle \tag{2.16}$$

where $F_i, U_{i<x>}, U_{i<y>}$ represent the pull force, the x- and y-coordinate of the i^{th} matching unit and k indicates the number of units which are considered. Usually using U2 and U3 comply with the calculation but in some cases (see figure 2.4(b)) U4 has to be considered.

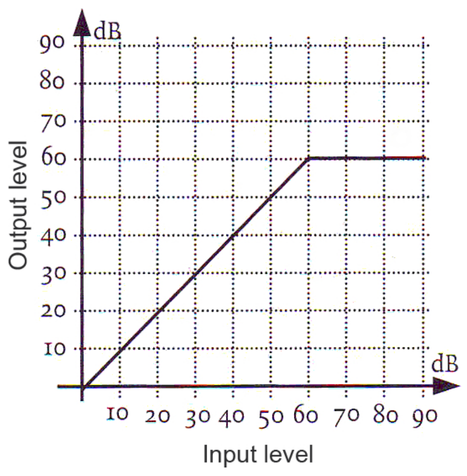
My own contribution to the PlaySOM, an acoustic evaluation tool called cLynx, collects the resulting spatial coordinates, which form the basis for the creation of the spatial mix-down of the Distance Mixer which is discussed in section 3.2.3.

2.3 Control Amplifiers

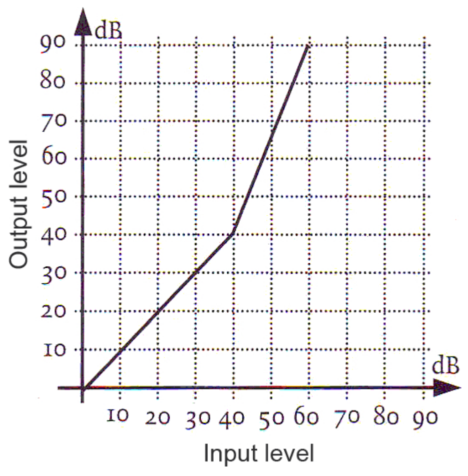
When listening to several different audio tracks it often seems that the songs strongly vary in their volume level. Especially tracks of the genres Pop, Rock and Heavy Metal seem to be very loud in contrast to classical pieces. A reason for this can be found in the difference



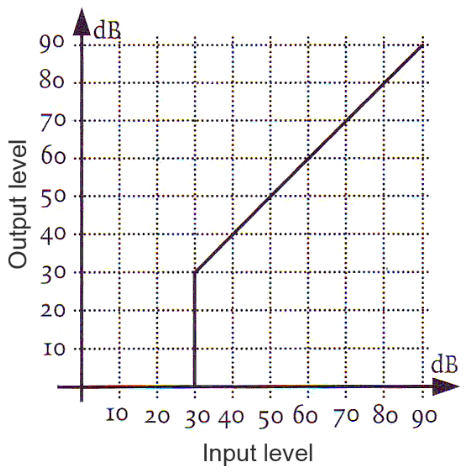
(a) The Compressor decreases high levels.



(b) The Limiter limits the signal to a defined threshold without influencing the signal below.



(c) The Expander increases high levels.



(d) A Noise Gate suppresses all signals falling under a defined threshold.

Figure 2.5: Static characteristic of the basic Control Amplifier types [Raffaseder, 2002].

between the most quiet and loudest volume level of an audio signal called the dynamic range. In order to approximate all volume levels, the dynamic range of each song has to be adapted.

Control Amplifiers are used in digital audio signal processing in order to amplify an input signal against the amount of its level. In contrast to other amplifiers they are not based on definitely but level- and time-dependent varying amplification. According to their varying functionality [Henle, 2001] distinguishes between:

- Compressor
- Limiter

- Expander
- Noise-Gate

The relation between the input and output signal for the different amplifiers can be illustrated by the so called static characteristic. Figure 2.5 illustrates the main Control Amplifier types along with their corresponding characteristic. The dashed lines describe an amplifier without amplification. Control Amplifiers are used in the following situations:

- Original dynamic is greater than the possible dynamic range of an audio device
- Over-modulation prevention
- Noise suppression
- Making a mix-down[2] more powerful and compact
- Sound-design
- Volume increase

2.3.1 Dynamic

Control Amplifiers modify the dynamic of an audio signal. The term dynamic is defined as the difference between the highest and lowest sound pressure level (SPL) of a signal also referred to as sound level. The sound level L_p is defined as:

$$L_p = 20 \lg \frac{p}{p_0} \tag{2.17}$$

The SPL corresponds to the 20^{th} logarithmic relation of a sound pressure p to a covenant reference sound pressure p_0. The reference pressure p_0 nearly defines the human auditory threshold for a 1000 Hz frequency:

$$p_0 = 2 * 10^{-5} Pa \quad (= 2 * 10^{-4} \mu bar) \tag{2.18}$$

Figure 2.6 displays the auditory and pain threshold in dependency of the frequency. Resulting from the difference between the auditory threshold and the pain threshold in the area of 1 kHz and 2 kHz the human ear is able to sense a maximal dynamic range of about 120 dB to 130 dB.

Regarding the point of dynamic measurement [Dickreiter, 1997] and [Warstat and Goerne, 1994] differ between:

- Original dynamic
- Program dynamic

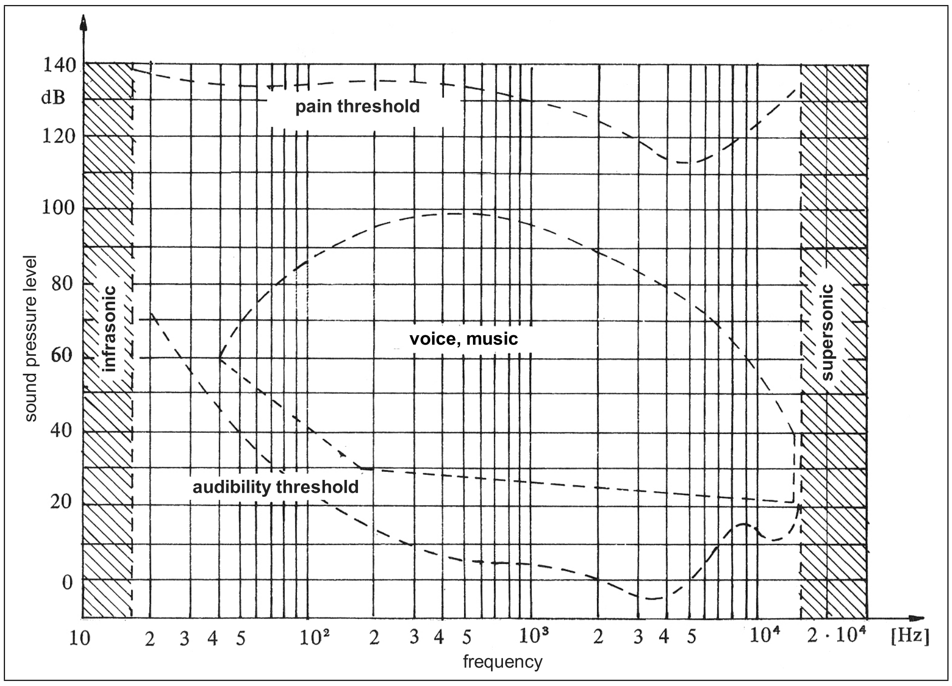


Figure 2.6: The human's audibility limit in decibel in dependency of the frequency [Dickreiter, 1997].

Sound Source	**Dynamic Range**
Human voice (no whispering and screaming)	15 dB – 20 dB
Vocal solo (classical music)	40 dB – 50 dB
Bowed instruments	30 dB – 35 dB
woodwinds	30 dB – 35 dB
clarinet	40 dB – 50 dB
brass instrument	40 dB – 50 dB
piano	40 dB – 50 dB
symphonic orchestra	
small instrumentation	30 dB – 40 dB
medium instrumentation	40 dB – 50 dB
big instrumentation	50 dB – 70 dB

Table 2.1: Estimated guide values of the dynamic range of the human voice, several classes of instruments and orchestras.

Original and Program Dynamic

The original dynamic describes the dynamic of the source signal and therefore is provided by an instrument or the human voice. Table 2.1 illustrates averaging, estimated original

[2]combination of multiple audio components into a final track

dynamic values of different sound sources. Although music could cover an enormous dynamic compared to the human ear in praxis a dynamic of such dimension is confined to natural bounds and therefore unwanted. [Warstat and Goerne, 1994] describes those borders based on the following example:

As illustrated in table 2.1 a symphonic orchestra represents a very dynamic sound source with a dynamic range of about 60 dB between the pianissimo[3] and fortissimo[4] passages. This kind of music normally is performed in very quiet environments (concert hall, opera etc.) which have an averaging background noise in the region of about 20 dB. Under normal circumstances the sound pressure level in the most quiet parts could be 40 dB and of the loudest parts 100 dB. So the pianissimo passages are 20 dB above the background noise and therefore clearly hearable for the visitors of the concert hall.
When recording this dynamic on a portable device and playing it back on the home hi-fi system you are faced with the following situation: The averaging background noise of an apartment is in the region of about 40 dB. In order to hear very quiet parts in a comfortable way those passages should lie 20 dB above the background noise at about 60 dB. Then, the loudest passages would have a level of 120 dB if the hi-fi system and loud speakers support that.

The example shows that a great dynamic range could not be utilized by the listener in a normal every day life situation because pianissimo parts are not hearable and fortissimo parts are too loud. Therefore, the original dynamic range has to be reduced conveniently so that it is suitable in the privacy of one's home. This reduced original dynamic is called the program dynamic. According to the example above the program dynamic is the dynamic which is presented to the listener by the hi-fi system. As an extreme case of program dynamic usually the car stereo is mentioned. The car's background noise depending on the velocity and car type is in the region of about 60 dB to 100 dB. In such a case a program dynamic of 10 dB to 20 dB seems to make sense. In contrast to the situation in a car a dynamic of up to 90 dB is possible in cinemas.

Table 2.2 presents the technically achievable dynamic on several data storage devices. This device-related dynamic is called the maximal system dynamic. Both maximal system dynamic and the above explained natural bounds make a convenient compression of the original dynamic necessary. This compression could be performed manually by an audio engineer or automatically by using Control Amplifiers such as Compressors or Limiters.

2.3.2 Compressor

A Compressor is a Control Amplifier, which reduces the original dynamic to a predefined program dynamic. As described in section 2.3.1 the compression of the dynamic of the

[3]Pianissimo or pp- means very quiet parts of a piece of music.
[4]Fortissimo or ff- means very loud parts of a piece of music.

Audio Device	**Dynamic range**
Professional analog Mixer	> 90
Analog tape	60 dB
UKW radio	66 dB
Gramophone record	61 dB
Video Home System (VHS)	55 dB
Audio CD and DVD	90 dB

Table 2.2: Dynamic range of several studio devices.

original signal is necessary because of several reasons. To achieve a dynamic reduction the level carefully has to be re-adjusted by a sound engineer or electronic device when the volume changes. According to this a Compressor can be seen as an automatic control, which starts its action when a signal exceeds a defined threshold. When an input signal exceeds the threshold it is attenuated by the Compressor. The dimension of attenuation is controlled by a parameter called compression ratio. [Henle, 2001] describes this parameter using the following example:
When a signal exceeds a defined threshold the Compressor reduces the signal using a fixed compression ratio. A compression ratio of 4:1 means that a 4 dB increase of the input signal only causes a 1 dB increase for the output. For signals which do not reach the threshold the Compressor uses a compression rate of 1:1 which does not cause a reduction.

Figure 2.7 graphically illustrates the operating mode of the compression ratio using a threshold of -20 dB. If the compression ratio is increased to about 20:1 the Compressor is called a Limiter. An output signal nearly never exceeds the threshold of a Limiter. For strict mathematical purpose a Limiter must have a compression ratio of ∞:1 to do that but in praxis the use of a ratio of about 20:1 creates satisfying results. Therefore, when increasing the compression ratio ρ you can differ between several kinds of Compressors:

$$\text{Compressor: } \rho =]1, 9] : 1$$
$$\text{Limiter: } \rho = [10, 90] : 1$$
$$\text{Clipper: } \rho = [91, 100) : 1$$

Two other important parameters of a Compressor are represented by the attack- and release time. They define the time a Compressor needs to take effect after a signal exceeded or fell under the threshold. The attack time defines the time the Compressor has for a stepwise increase of the compression till the full compression ratio is reached. In contrast to the attack time the release time defines the time a Compressor has for a stepwise reduction of the compression till the compression has no further effect. A too short attack time causes disturbing clicking noise because nearly every time the signal exceeds the

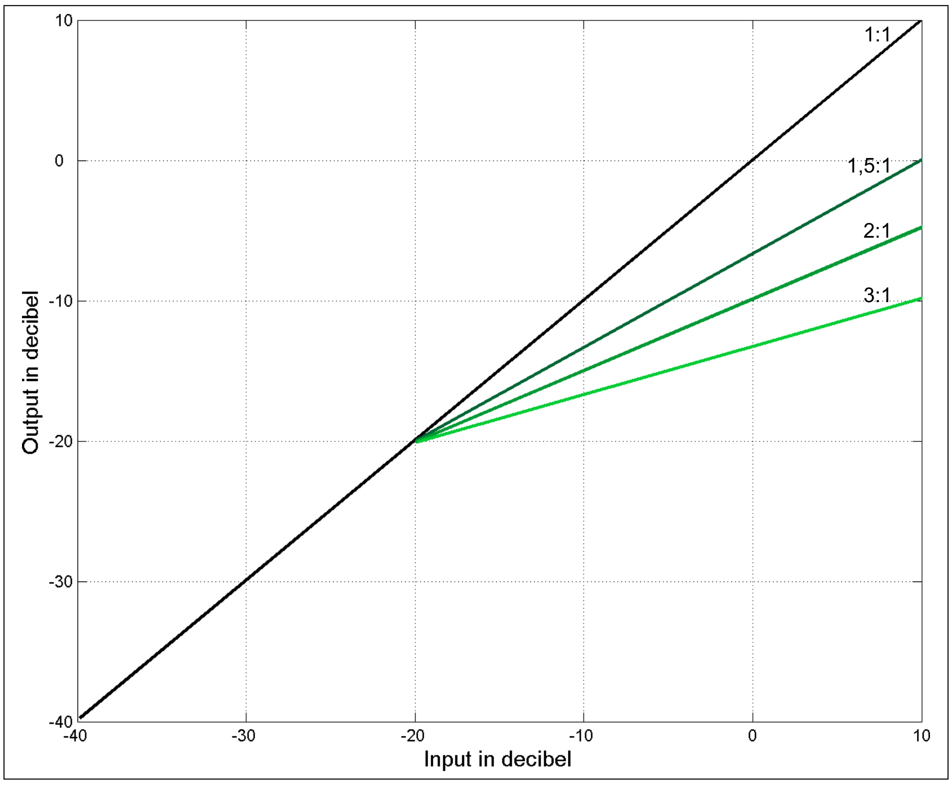


Figure 2.7: Several compression ratios and their effect to the output signal.

threshold the maximal compression takes effect. Therefore, the attack time should be defined in the dimension of covering several periods of the basic oscillation. Also the release time should not be defined too short. If so every fall under the threshold would immediately lead back to the original signal. [Henle, 2001] suggests an attack time of 50 ms to 100 ms for a low tonic keynote and 10 ms to 50 ms for a high tonic keynote. The release time in general should be defined at minimum 200 ms.

The compression of the sum of signals (recorded music) as performed by the cLynx tool happens in the same way as the compression of a single instrument. A smooth constant compression is advised by [Henle, 2001] using a compression ratio of 2:1, an attack time of 50 ms to 70 ms and a release time of 200 ms. In order to work against one sided compression of stereo signals, two identical Compressors are needed, which additionally interact together. Otherwise, the use of a Compressor could result in balance displacement.

In many cases the input signal is filtered in order to compress only specified parts of the signal. This is called frequency dependent compression as the compression is not equal for every frequency. Such frequency dependent Compressors and Limiters are better known as DeEsser or Multiband-Compressor.

2.3.3 Limiter

A Limiter works in the same way as a Compressor. It has the same parameters but a compression ratio of at least 10:1. So the Limiter sets all values exceeding the threshold to the threshold level. Therefore, a Limiter is used in situations where a level is not allowed to exceed a defined threshold. E.g., such a situation is the digital recording which becomes damaged when exceeding the maximal possible oscillation. Additionally, it seems to be clear that the attack- and release time should be chosen very small.

2.3.4 Expander

The Expander represents the direct opposite of the Compressor. While the Compressor makes low levels subjectively louder by reducing high levels, the Expander sinks low levels. As soon as the level of a signal falls under a defined threshold it is reduced using a fixed ratio. Because signals with low levels are reduced stronger the dynamic of the output expands. The ratio normally is in the region of 1:2 to 1:8. Like the Compressor the Expander can define the time parameters attack and release time.

2.3.5 Gates

A Gate is a special kind of Expander with a high ratio of 1:10, 1:20 and more. That high ratio causes that a signal which falls under the threshold is completely suppressed. Gates are used to hide noise and disturbing signals. Therefore, Gates are often called Noise Gates as well. In analogy to the Expander the Noise Gate defines all important parameters such as ratio, attack and release time.

2.4 Evaluation of Acoustic Similarity

The increasing availability of music libraries requires new ways to interactively let users explore music repositories. The concept of detecting similar tracks raised its importance as an aid for easy navigation and instant retrieval. Systems which are based on this concept, namely Music Similarity Systems , assist the user in finding new music that is musically most similar to a defined query song according to a defined similarity measure. Several approaches of similarity measures have been published, e.g. [Foote, 1997], [Pampalk et al., 2002], [Tzanetakis et al., 2001], all containing performance results of disparate evaluation processes. However, the evaluation of such methods is difficult because [Pampalk, 2006]:

- most implementations are not freely available
- copyright restrictions do not allow sharing the audio data used for the evaluation
- the use of different music data sets produces incomparable results

which increased the demand of a standardized evaluation within the Music Information Retrieval (MIR) community.

This section is focused on human evaluation processes of Music Similarity Systems which are an especially difficult task, because acoustic music evaluation inherently is biased by subjective cognition, interpretation and reasoning which usually is based on knowledge and conventions of the real world. Furthermore, psycho-acoustic research has found out that parameters like tempo and genre information have major impact to similarity judgments [Cupchik et al., 1982].

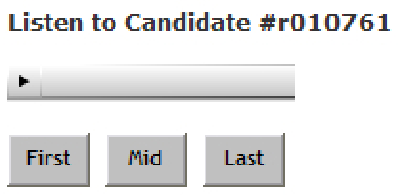
2.4.1 MIREX

In recent years there has been a strong effort to standardize evaluations within the MIR community. Therefore, various evaluation contests have been arranged within several International Conferences on Music Information Retrieval (ISMIR[5]) since ISMIR 2004. During the 6th ISMIR Conference in 2005 (London) the 1st dedicated evaluation project, called Music Information Retrieval Evaluation eXchange (MIREX), was conducted which aims at comparing state-of-the-art algorithms and systems relevant for Music Information Retrieval. All evaluation tasks and results of all MIREX contests are published on the MIREX website[6]. In 2007, MIREX attempted a large scale music similarity evaluation as part of two of its tasks for the first time which primarily relied on human judgment to rank the various systems. Furthermore, a correlation between objective measures based on genre and artist classification and the human evaluations has been detected.

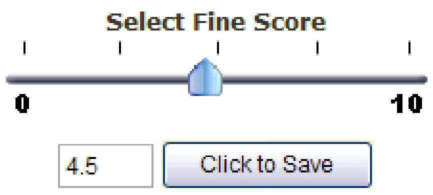
The contribution of Stephen J. Downie's IMIRSEL lab to the subjective evaluation of the MIREX 2006 called Evalutron 6000 (E6K) is discussed in [Gruzd et al., 2007]. E6K is a web-based system which supported human judges in evaluating algorithms submitted to the 'Audio Music Similarity and Retrieval' and 'Symbolic Melodic Similarity' tasks by logging their measuring of similarity sensation according to query-candidate pairs through interaction dialogs. Each query and candidate song was divided into three parts covering the first, middle and last 30 seconds of every track. Clicking one of the buttons displayed in figure 2.8(a) started playback of the selected song part. After being aware of the song pair the participants were asked to rate similarity by a coarse score (Not Similar, Somewhat Similar and Very Similar) using the dialog depicted in figure 2.8(c). Additionally, a fine score - a number between 0.0 (Least similar) and 10.0 (Most similar) - had to be specified using the dialog of figure 2.8(b). For MIREX 2006 E6K collected all user-interactions with the system which consisted of 69.745 judgments. While E2K forced the subjects to listen to the results song per song, the cLynx evaluation tool presented in this thesis also enables

[5]ISMIR is derived from International Symposium on Music Information Retrieval which was redefined to International Conference on Music Information Retrieval between ISMIR 2001 and ISMIR 2002.

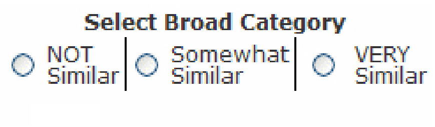
[6]MIREX web page is available at http://www.music-ir.org/mirexwiki/index.php



(a) Several play buttons for the query song.



(b) Coarse category selection buttons.



(c) Fine score selection scale.

Figure 2.8: Evalutron 6000 GUI components (source: MIREX web page).

the playback of all result songs simultaneously or sequentially with different emphasis.

2.4.2 Subjective Evaluations

Several user studies have been carried out by individual research groups in order to evaluate similarity measures. Besides evaluating the system's usability, which is based on the efficiency a user performed a given task and partly on questionnaires, humans were invited to evaluate the similarity measure of several systems. All evaluations which are discussed in the following paragraphs were accomplished by listening tests.

A simple evaluation involving only two human evaluators is documented in [Logan and Salomon, 2001]. Logan and Salomon presented several play lists of 20 tracks generated from their system by using one query song to two independent respondents, each being asked to rate every track in the play lists as 'similar' or 'not similar' to the query song. The test revealed a high agreement between the users by only rating 12% of the songs differently. The results of [Pampalk, 2006] confirm this consistency between listeners. Through comparing the average number of similar songs for the first 5, 10 and 20 songs to random results showed that their measurement works significantly better.

Another similar subjective evaluation is published in [Aucouturier and Pachet, 2002]. While Logan and Salomon only involved 2 users for their evaluation, Aucouturier and Pachet conducted the study with 10 users from their lab. Each user was presented a seed song S and two test songs A and B. The test songs were chosen in a way that the distance $\overline{SA}$ calculated by their similarity measure was greater than the distance $\overline{SB}$. The users had to rate the similarity of both test songs to the seed song and agreed in 80% with the ordering of the similarity measure.

In the listening test of [Allamanche et al., 2003] a similarity measure was compared to random. 20 target songs were selected randomly and their five most similar tracks and their worst similar track were computed. Additionally to these six songs a randomly chosen track was presented to 10 participants which had to rate the similarity of each song to its corresponding seed. The test pointed out that their similarity measure works better than random.

Vignoli and Pauws published an application based evaluation [Vignoli and Pauws, 2005] in which they tried to reason the usability of their system and the performance of their similarity measure in comparison to two control systems. A task was given to 22 participants who had to perform a task on each system. The participants' job consisted of creating an appropriate play list according to a defined everyday life situation by using the systems. They detected that more explorative users were able to create better play lists in less time and less effort when using their system.

2.5 Summary

This chapter introduced unsupervised learning techniques relevant for music segmentation. Besides different approaches for dividing a set of observations into a specified or unspecified number of groups based on various similarity measures, some dimension reduction methods are explained, which decrease the dimensionality of data while retaining as much as possible of the data's information. Both techniques are applied to an Artificial Neural Network approach, called Self-Organizing Map, which is used by the application PlaySOM in order to structure the underlying music collection. Because the major difficulty of evaluating systems, which perform music organization on the basis of similarities measures, results from the individuality of human cognition, it makes sense to evaluate the quality of such applications based on listening tests. It is important to present a meaningful and clear mix-down to the subjects, which is obtained by applying Control Amplifiers, which combine the reduction of the dynamic range and the smoothing of the dynamic of audio signals. Based on the components of this chapter the cLynx tool assists users in judging music similarity and is presented in the next chapter.

Chapter 3

cLynx: A Novel Form of Music Similarity Evaluation

3.1 Overview **30**
3.2 Components **30**
3.2.1 Analysis Component 32
3.2.2 Normalization Component 34
3.2.3 Mixer Component 38
3.2.4 Extending cLynx 50
3.3 Run-Time Modes **51**
3.3.1 Stand-Alone Mode 52
3.3.2 Console Mode 53
3.3.3 GUI Mode 53
3.3.4 Plug-In Mode 54
3.4 Summary **55**

In this chapter the cLynx application is discussed which was implemented in the course of this thesis. cLynx aims at supporting users in acoustic evaluations of audio-based music similarity measures. The application offers the evaluators several different approaches for accessing the underlying data of similarity systems in order to help them discovering, analyzing and evaluating cluster structures of audio data much faster and more efficiently. Several different types of mixers assist the evaluators in exploring the data in order to get a meaningful overview of the underlying data's structure which allows them to detect whether a region consists of homogeneous or heterogeneous data and also whether it contains outliers.

3.1 Overview

The cLynx tool is provided as the package clynx.jar implemented in Java version 6.0 but also available for Java version 5.0. It provides techniques for acoustically evaluating cluster structures. To ease listening to the sound for the evaluator the package additionally offers three different methods for analyzing and three different methods for normalizing the audio content besides the currently five different methods of mixing the audio signals to a single output stream. These three steps define the three encapsulated main parts the itinerary of cLynx is divided in:

- Analysis
- Normalization
- Mixing

These parts are documented in more detail in section 3.2.

Section 3.3 presents different modes the application can be executed in, which for instance are defined by the execution command passed to the console. For an easier execution example start files for Linux and Windows systems are stored in the application directory which start cLynx in one of the two possible Stand-Alone Modes (see section 3.3.1). For developers of Music Information Retrieval (MIR) systems an interface is specified which allows them to integrate the plug-in version of the application into their system, described in section 3.3.4.

First of all the itinerary including the three main parts of the application is discussed in detail, followed by an explanation of the different modes in which cLynx can be started.

3.2 Components

Depending on the run-time mode cLynx is executed in, cLynx's application flow comprises of three to four main steps which are depicted in figure 3.1. Apart from the Stand-Alone Console Mode which is explained in section 3.3.2 the applications flow consists of the steps: Analysis, Normalization, Mixing.

If cLynx extends an external application as plug-in all required parameters are passed from the underlying system. In the Stand-Alone GUI Mode all parameters are defined by user interaction through several dialogs. Therefore, these run-time modes do not require a validation of the input parameters, but directly start in step two, the analysis. If cLynx is executed in the Stand-Alone Console Mode the required parameters must be passed through a properties file which is processed in the first step *Input Validation.* The parameters of the properties file are listed in table A.1. As depicted in figure 3.1 cLynx in this mode can only reach step two when passing the validation step.

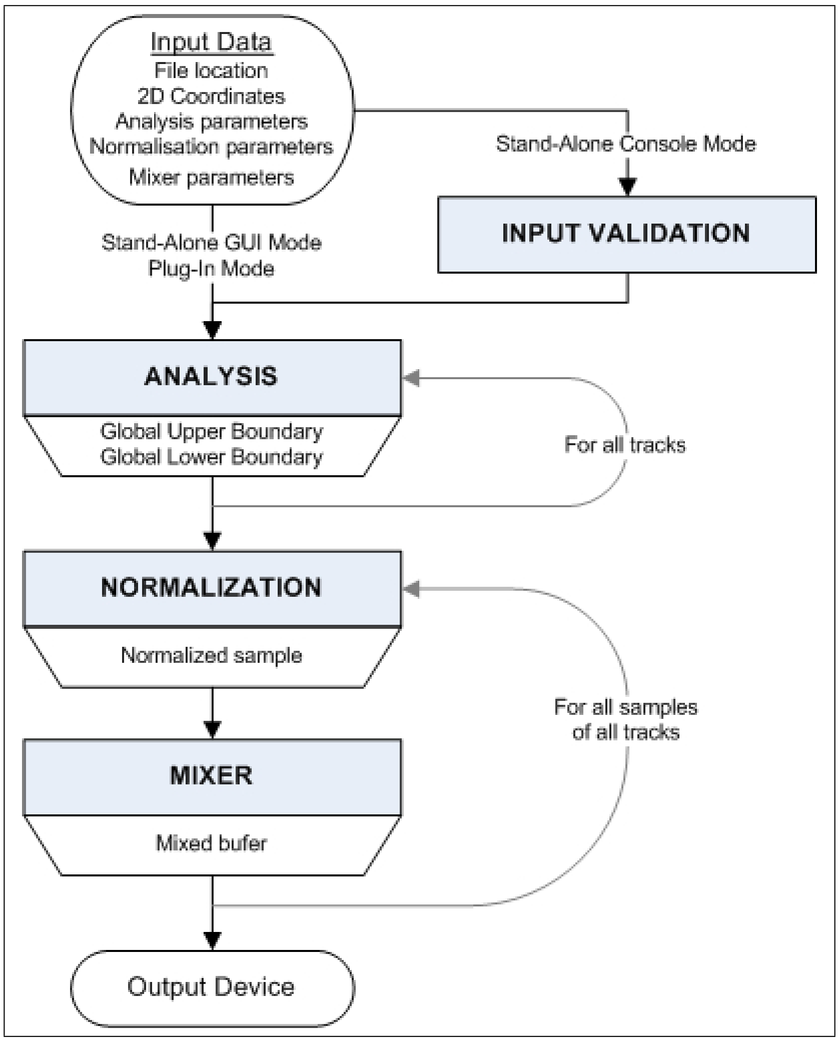


Figure 3.1: cLynx's itinerary: Depending on cLynx's run-time mode the application flow consists of three or four steps.

The *Analysis* step entirely processes all passed sound files and calculates depending on the signals' sample values a global upper and lower boundary value for all. The calculation of these boundaries differs between the various kinds of analysis components and is discussed in section 3.2.1. Therefore, an analysis component has to be defined from the user before the execution of the application. As shown in figure 3.1 the global boundary values are passed to the succeeding step *Normalization* which is explained in detail in section 3.2.2

The passed set of audio files usually contains tracks of several genres in varying record quality which results in strongly differing volume levels between the tracks. Therefore, all signals are normalized. The normalization process tries to reduce the dynamic range of all signals in order to get nearly equal volume levels for all pieces of music. In contrast to the analysis step the normalization component is not a secluded process, but directly interacts with the mixer process. All samples are normalized in run-time directly before they are sent through the mixer to the output device. This permits changing several normalization parameters in run-time.

The *Mixer* step represents the core of the application. As displayed in figure 3.1 the mixer process multiplexes all normalized sample values to one single output stream. Because cLynx is designed for empirical evaluation and the human sense individually differs several mixers are implemented which take this issue from three different angles, as explained in section 3.2.3.

According to the application flow in this section the analysis component is explained first, followed by the normalization process. In the last subsection the mixer component is discussed.

3.2.1 Analysis Component

The analysis component represents the most important precondition for presenting a proper and clear output signal to the listener for his acoustic evaluation. Only a well mastered composition of the multiple input signals which additionally takes into account the structure of all other audio tracks can ensure an effective and fast evaluation. While the mastering process is performed by the normalization component the analysis process is responsible for the calculation of the required values for the normalization, an upper and lower sample value indicator called the global upper and lower boundary. The global boundaries define the range of the program dynamic the original dynamic of the signals is mapped to by the normalization process, as discussed in section 2.3.1. The basic values for the global boundary calculation represent the so called description values. One set of description values characterizes the sample structure of a single audio signal and contains the following values:

- Minimal Sample Value
- Maximal Sample Value
- Arithmetic Mean Value
- Root Mean Square Value
- Standard Deviation Value

In order to compute a set of description values an audio file has to be processed entirely which is a quite time-expensive task. E.g., a 16 Bit, stereo PCM signal of 1 MB contains about 10^6 Byte data. One sample value needs 2 Bytes to be stored which furthermore means that the given signal is represented by 500.000 samples which have to be processed by the analysis component. If then S_i is the set of all samples of a file i and Θ_i is its set of description values the minimal sample value $\Theta_{i,min}$ and the maximal sample value $\Theta_{i,max}$ are defined as:

$$\Theta_{i,min} = \min(S_i) \tag{3.1}$$

$$\Theta_{i,max} = \max(S_i) \tag{3.2}$$

If S_i contains n elements s the arithmetic mean value $\Theta_{i,\mu}$ and the root mean square value $\Theta_{i,rms}$ are defined as:

$$\Theta_{i,\mu} = \frac{1}{n}\sum_{k=1}^{n} s_k \qquad s_1, ...s_n \in S_i \tag{3.3}$$

$$\Theta_{i,rms} = \sqrt{\frac{1}{n}\sum_{k=1}^{n} s_k^2} \qquad s_1, ...s_n \in S_i \tag{3.4}$$

In order to have a measure of the distribution of the samples within the given signal the standard deviation $\Theta_{i,\sigma}$ is calculated by using the arithmetic mean, defined as:

$$\Theta_{i,\sigma} = \sqrt{\frac{1}{n}\sum_{k=1}^{n}(s_k - \Theta_{i,\mu})^2} \qquad s_1, ...s_n \in S_i \tag{3.5}$$

Especially the calculation of the standard deviation is very time-expensive because σ can only be calculated in processing the signal twice because the mean value is needed for its calculation which is computed in the first run together with the min/max values. The description value set Θ of every file is stored in the application until its next execution in order to save time when one audio signal occurs several times during the evaluation.

To accelerate the analysis process parallel analysis of the signals is supported but only available in the Stand-Alone Console Mode because tests pointed out that analyzing more than three audio files in parallel often leads to a Java Heap Exception. Another approach to raise the computational performance of the analysis component deals with calculating the set of description values from only a predefined number of sections of an audio file, which is discussed in section 4.2.

The audio signals' sets of description values build the basis for the calculation of a global upper and lower boundary. Depending on the kind of analysis component the calculation of the global boundaries is different as discussed in detail in the next paragraphs. The resulting boundary values are passed to the next step, the normalization component.

MinMax Analyzer

As the name already indicates the minimal sample values Θ_{min} and maximal sample values Θ_{max} of all audio signals represent the basis for the calculation of the global upper and lower boundary for the MinMax Analyzer. Because it is unwanted to expand the dynamic range of not even one signal, as explained in section 3.2.2, the global upper boundary is represented by the minimal Θ_{max} and the global lower boundary by the maximal Θ_{min} of all audio signals. If then B_{top} is the global upper boundary and B_{bot} is the lower boundary they are defined as:

$$B_{top} \leftarrow \min(\Theta_{max}) \tag{3.6}$$

$$B_{bot} \leftarrow \max(\Theta_{min}) \tag{3.7}$$

Statistical Analyzer

The Statistical Analyzer uses the statistical values of the description sets to calculate a global upper and lower boundary. It calculates the boundaries either by using the arithmetic mean or the root mean square. Therefore, for every sound file i a temporary upper boundary value is defined, e.g. for the arithmetic mean case as $max_i = \Theta_{i,\mu} + \Theta_{i,\sigma}$ and a temporary lower boundary value as $min_i = \Theta_{i,\mu} - \Theta_{i,\sigma}$. If then M_{top} is the set of all temporary upper boundary values max_i and M_{bot} the set of all temporary lower boundary values min_i the global upper boundary B_{top} and global lower boundary B_{bot} are defined as:

$$B_{top} \leftarrow \min(M_{top}) \tag{3.8}$$

$$B_{bot} \leftarrow \max(M_{bot}) \tag{3.9}$$

3.2.2 Normalization Component

When listening to several different audio tracks it often seems that the signals strongly differ in their volume. Especially tracks of the genres Pop, Rock and Heavy Metal seem to be very loud in contrast to classical tracks. A reason for this can be found in the difference between the most quiet and loudest volume level of an audio signal called the dynamic range which measures the original dynamic of an audio signal as discussed in section 2.3.1. Therefore, the dynamic range Δ_{orig} of the signal i is defined as:

$$\Delta_{i,orig} = [\Theta_{i,min}, \Theta_{i,max}] \tag{3.10}$$

For example, tracks of the genre Heavy Metal normally have a very small dynamic range because most of the Heavy Metal tracks start with high volume and continue with that volume level until the end. In contrast to that a classical piece of music has a very high dynamic range because it often contains very quiet parts followed by acutely loud parts. That is why a classical track often is called high dynamical as well in contrast to a Heavy

Metal piece of music which is often defined as low dynamical. Imagine playing a metal track followed by a classical track. When the main volume control was reduced because of the loud metal track the quiet parts of the succeeding classical track will become inaudible. To take a step forward, imagine the same scenario using parallel audio playback. In that case the Heavy Metal signal usually will be so dominant that nearly the whole classical piece of sound will not be hearable. To avoid that problem all signals are normalized before they are mixed to one single output signal in the mixer component. The normalization process tries to combine the reduction of the dynamic range and the smoothing of the dynamic which is explained in section 2.3.1. Only when the dynamic range and the dynamic of all audio signals are nearly equal the volume will not strongly differ. Therefore, during the normalization process N the volume level of every single audio signal is modified in a way, so that all resulting signals nearly have the same volume level. That means, the signals' original dynamic is transformed to a program dynamic (see section 2.3.1). The range of the program dynamic Δ_{prog} is equal for all signals and is defined as:

$$\Delta_{prog} \quad = \quad [B_{bot}, B_{top}] \tag{3.11}$$

B_{bot} and B_{top} define the global upper and lower boundaries and were calculated by the analysis component which is explained in section 3.2.1 in order to do that transformation every sample value s from the original dynamic space is mapped to a value s_{norm} of the program dynamic space, defined as:

$$N(s) \quad = \quad s_{norm} \tag{3.12}$$

According to the type of the selected normalization component N the sample value s is mapped in a linear or dynamic way. Currently, a Linear Amplitude Normalizer and a Compressor are implemented and described in the following paragraphs. The extension to new additional normalizations is easily enabled. After the successful normalization all s_{norm} values are passed to the mixer step.

Linear Amplitude Normalization

The Linear Amplitude Normalization modifies an audio signal by scaling the current sample value s proportional to the ratio of the range of the program dynamic to the range of its original dynamic. If then S_i specifies the set of all sample values of an audio signal i and B_{top} and B_{bot} define the upper and lower bounds of the program dynamic s_{norm} is defined as:

$$s_{norm} \leftarrow \frac{B_{top} - B_{bot}}{\Theta_{i,max} - \Theta_{i,min}} * s \qquad s \in S_i \tag{3.13}$$

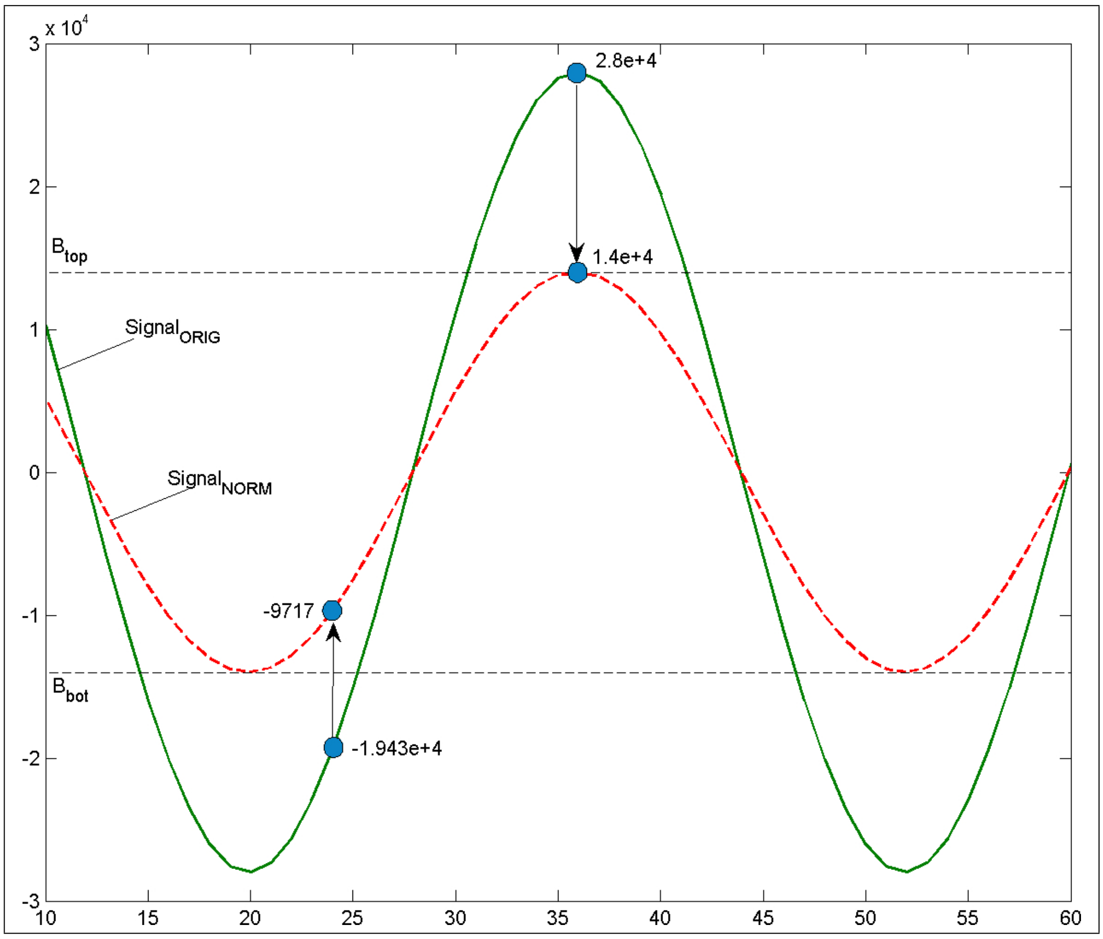


Figure 3.2: Linear mapping of a Linear Amplitude Normalization which maps the sample values of a signal with a dynamic range of $[2.8e^4, -2.8e^4]$ into the range $[1.4e^4, -1.4e^4]$.

As the name already describes Linear Amplitude Normalization performs a linear mapping of the sample values which means that very loud values are normalized in the same way as quiet values. It maps the whole signal into the defined range $[B_{bot}, B_{top}]$ resulting in the equal dynamic range for every normalized audio signal. Therefore, the linear mapping causes a modification of the range of the signal's dynamic but does not modify the dynamic itself. If $\{s_1, s_2, \ldots, s_n\}$, $s_j \in S$ is the set of all sample values of an audio signal the ratio of every sample s_j to its proximate sample s_{j+1} does not change:

$$\frac{s_j}{s_{j+1}} = \frac{s_{norm,j}}{s_{norm,j+1}} \tag{3.14}$$

Figure 3.2 describes the scenario of the Linear Amplitude Normalization graphically for a signal with a original dynamic range $\Delta_{orig} = [2.8e^4, -2.8e^4]$ which is mapped to a program dynamic with a range $\Delta_{prog} = [1.4e^4, -1.4e^4]$.

Compressor

The Compressor performs a nonlinear mapping of a signal from its original dynamic space to a defined program dynamic space. In contrast to the Linear Amplitude Normalization a Compressor additionally not only takes the dynamic range but also the dynamic of an audio signal in account. As discussed in the previous paragraph the Linear Amplitude Normalization reduces the dynamic range while not changing the dynamic (see section 2.3.1) of the signal. In order to improve the listening comfort for the evaluator, the dynamic of every single audio signal must be adjusted. As described in section 2.3.2 a Compressor only maps samples of a signal to the program dynamic space which match a defined precondition. When a sample value is outside the range of the program dynamic Δ_{prog} the sample value, which exceeds the threshold, is modified by using a so called *compression ratio* ρ. Therefore, the normalized value s_{norm} of the sample s is defined as:

$$s_{norm} \leftarrow B_{top} + \rho * |B_{top} - s| \qquad s > B_{top} \tag{3.15}$$

$$s_{norm} \leftarrow B_{bot} - \rho * |B_{bot} - s| \qquad s < B_{bot} \tag{3.16}$$

$$s_{norm} \leftarrow s \qquad s \in \Delta_{prog} \tag{3.17}$$

In order to suppress that the Compressor takes effect by using the full compression ratio after a sample exceeded or fell under the boundaries a time parameter is included called attack time which is discussed in more detail in section 2.3.2. The attack time defines the time the Compressor has for a stepwise increase of the compression till the full compression ratio is reached. In contrast to the attack time another parameter is specified called release time which defines the time a Compressor has for a stepwise reduction of the compression till the compression has no further effect which is applied when an exceeded sample value is followed by an value which lies in the range Δ_{prog}. If then t specifies the time the samples lie outside the given range Δ_{prog} the normalized sample value s_{norm} is defined as:

$$s_{norm} \leftarrow B_{top} + \rho * \min\left(\frac{t}{t_{attack}}, 1\right) * |B_{top} - s| \qquad s > B_{top} \tag{3.18}$$

$$s_{norm} \leftarrow B_{bot} - \rho * \min\left(\frac{t}{t_{attack}}, 1\right) * |B_{bot} - s| \qquad s < B_{bot} \tag{3.19}$$

$$\tag{3.20}$$

The Compressor performs a nonlinear mapping which results in an approximation of all samples $s \notin \Delta_{prog}$ to the boundaries B_{top} and B_{bot} proportional to the defined ratio ρ. Therefore, the range of the program dynamic of every single audio signal is usually unequal to the others but within an acceptable range. If S_i and S_j are the sets of all samples of the audio signals i and j the maximal difference δ between the upper boundaries of $\Delta_{i,prog}$ and $\Delta_{j,prog}$ is defined as:

$$\delta \quad \leftarrow \quad \rho * |\Theta_{i,max} - \Theta_{j,max}| \tag{3.21}$$

This illustrates that the divergence of the dynamic ranges is only influenced by ρ. Depending upon the increase of the ratio the divergence decreases.

In contrast to the Linear Amplitude Normalization the Compressor's nonlinear mapping additionally reduces the dynamic of the audio signal. Because it modifies the signal by reducing only sample values which are outside the range of Δ_{prog}, the ratio of every sample value s_j to its proximate sample value s_{j+1} might have changed after the compression process:

$$\frac{s_j}{s_{j+1}} \neq \frac{s_{norm,j}}{s_{norm,j+1}} \tag{3.22}$$

Therefore, a Compressor using a small range $[B_{bot}, B_{top}]$ but high ratio (e.g. $1 : 100$) could modify a high dynamical signal to a low dynamical signal. While the Linear Amplitude Normalization reduces the volume level of quiet parts in the same way as loud parts the Compressor only reduces the volume level of loud parts which makes quiet parts become more present.

3.2.3 Mixer Component

The mixer component represents the core of cLynx. It mixes all passed audio signals and multiplexes them to one single output signal. The mixer process is responsible to present the listener a clear and significant mix-down of all audio files which are part of the current empirical evaluation in a way that the user is able to create an acoustic association of a specified area in a cluster structure. According to the individuality of the human auditory sense and the differences in the person's ability of absorbing and filtering information several different mixer components are implemented. The mixer step tackles the issue from three different angles:

- sequential playback
- parallel playback
- playback using 2D information

Depending on the kind of mixer component the playback time, the comfort of hearing, the degree of concentration, the time to recognize all audio tracks etc. may differ. Maybe some mixer processes will allow evaluation over several hours while others will force the listener to pause the evaluation due increasing lack of concentration after already few minutes. All these facts are highly depending on the human individuality of the listeners. In section

4.3 an evaluation of the cLynx's applicability is presented which especially is focused on the individuality of the human sense.

After the mixer component has mixed the audio tracks the resulting signal is sent to the output device of the computer. A Clipper which is explained in section 2.3.2 is directly placed before sending the signal to the output device in order to avoid over-modulation of the output signal.

In the first part of this section two mixers are presented which follow the approach of sequential playback followed by two mixers which use parallel playback. In the last part a special kind of parallel mixer is discussed which presents the user a kind of spatial acoustic representation of the audio signals based on 2-dimensional spatial information.

Static Sequential Mixer

```
duration:=3.0
fadeTime:=0.4
current:=0

while size($\Phi$)>0 do {
    while duration has not expired do {
        //buffer contains information for all channels
        outputBuffer:=read($\Phi_{current}$)
        //sends the streamed data to the output device
        sendToDevice(outputBuffer)
    }

    while fadeTime has not expired do {
        //buffer contains the cross faded data
        outputBuffer:=crossover(read($\Phi_{current}$),read($\Phi_{current+1}$))
        sendToDevice(outputBuffer)
    }

    //removes all finished streams from $\Phi$
    cleanUp($\Phi$)

    current:=nextIndex()
}
```

Listing 3.1: The example illustrates how the Static Sequential Mixer works.

The Static Sequential Mixer merges all input signals sequentially into one single output signal. Every audio signal is played back alone on every available channel for a specified number of seconds defined by the *duration* parameter followed by another signal. The cross-over from one track to its successer can be performed by a soft or hard changeover which can be specified through the parameter *fadeTime* in seconds. The smaller *fadeTime*, the shorter the cross-over and the faster the proximate track is faded in. A *fadeTime* of 0 seconds disables the cross-over effect which results in immediately switching to the next track which often is conducted with a clicking noise.

Furthermore, the Static Sequential Mixer provides the best comfort of hearing because the listener is only faced with one single track. It can be assumed that the user is able to do the evaluation task by using that kind of mixer for quite a long time because he does not need to be highly concentrated to recognize the currently played back sound. However, the time the mixer needs in order to have played back every track at least once is very high compared to a parallel playback scenario.

Listing 3.1 presents in pseudo code the function of the Static Sequential Mixer. The *duration* and *fadeTime* parameters are initialized with 3.0 and 0.4 seconds. The array Φ contains all audio signals which are passed to the application. As long as Φ contains signals the playback is performed. In the first step the function *read()* reads a defined number of samples of the current element of Φ to a buffer called *outputBuffer*. After the reading process *outputBuffer* contains the data of all channels of one single track and sends this data to the output device by applying the function *sendToDevice()*. This procedure is repeated until the *duration* expires. Before playing back the next track a cross fade is performed by applying the function *crossover()*. The passed parameters are the read data of the current and succeeding track which again is done by the function *read()*. A further invocation of the function *sendToDevice()* sends the mixed data to the computer's output device. This procedure is repeated until *fadeTime* expires. After the function *cleanUp()* has removed all finished tracks from the array Φ and the control variable *current* was updated with the index of the next track the outer while-loop starts from the beginning.

Dynamic Sequential Mixer

The Dynamic Sequential Mixer creates an output signal by passing different input signals to the channels. That means, every channel contains a different audio signal in run-time which requires a conversion of all tracks to mono. The Dynamic Sequential Mixer enables a first kind of hearing tracks in parallel because the same number of tracks as channels are available is hearable. Nevertheless, it is ranked to the Sequential Mixer segment due its sequential playback for every channel. Figure 3.3 illustrates how the Dynamic Sequential Mixer works. After track T1 was played back on the first channel and track T2 was played back on the second channel for a defined number of seconds (*duration* parameter), T2 is

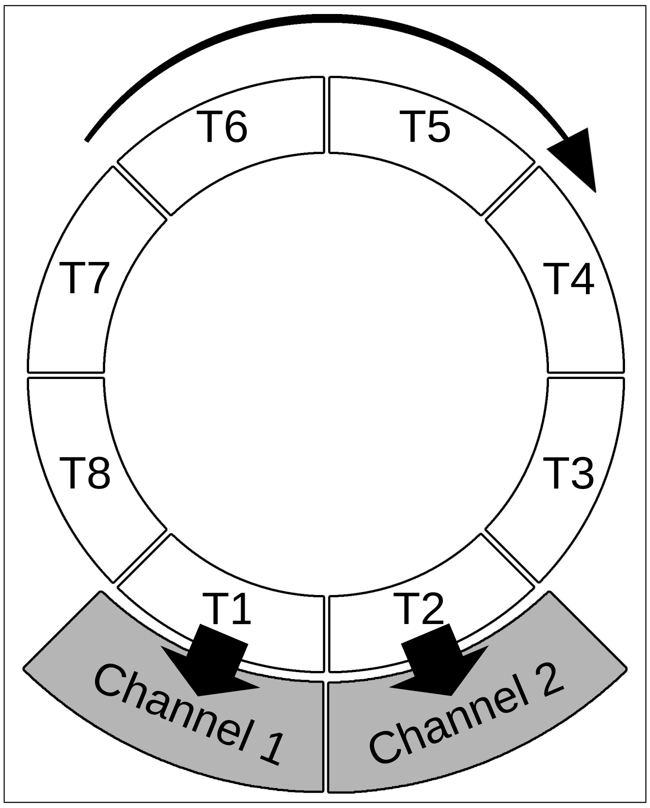


Figure 3.3: The Dynamic Sequential Mixer plays back one track per channel.

switched to channel1 and T3 to channel2. The changeover from T2 to the first channel, T3 to the second channel and the removing of T1 is performed by using fading techniques which have the following effects on the channels:

- channel1: T1 is faded out. The time for the fade-out is defined by *fadeTime*. Additionally, T2 is faded in which takes *fadeTime* seconds.
- channel2: T2 is faded out and T3 is faded in which both is performed in *fadeTime* seconds.

In contrast to the Static Sequential Mixer the time t the mixer needs in order to have played back every track at least once is shorter for the Dynamic Mixer. If t_{ssm} defines the time a Static Sequential Mixer needs to perform the full playback of all passed files and the parameter *channels* defines the number of active channels the time t_{dsm} variable for full playback of the Dynamic Sequential Mixer is specified as:

$$t_{dsm} = \frac{t_{ssm}}{channels} \tag{3.23}$$

Although the listener is faced with the same number of tracks within a shorter time which forces a higher degree of concentration a loss of overview of the currently played back tracks is not mandatory. It is still possible to recognize all currently active audio signals; even their lyrics can be identified. To ensure that an accurate normalization is necessary to minimize the possibility that one track drowns all others out.

```
duration:=3.0
fadeTime:=0.4
current:=0
channels:=2

while size($\Phi$)>0 do {
    while duration has not expired do
    {
        for i=0:channels {
            //reads the data for a single channel
            outputBuffer[i]:=read($\Phi_{current+i}$)
        }
        //sends the streamed data to the output device
        sendToDevice(outputBuffer)
    }

    while fadeTime has not expired do {
        for i=0:channels {
            //reads the data for a single channel
            outputBuffer[i]:=crossover(read($\Phi_{current+i}$),read($\Phi_{current+i+1}$))
        }
        sendToDevice(outputBuffer)
    }

    //removes all finished streams from $\Phi$
    cleanUp($\Phi$)

    current:=nextIndex()
}
```

Listing 3.2: Pseudo code for the Dynamic Sequential Mixer.

Listing 3.2 shows the function of the Dynamic Sequential Mixer. In contrast to the function of the Static Sequential Mixer which is explained in listing 3.1 the buffer *outputBuffer* represents an array of buffers. Every index specifies the data for one channel. The variables Φ, *duration* and *fadeTime* and the functions *read()*, *crossover()*, *cleanUp()* and *nextIndex()*

work in the same way as explained with listing 3.1. In the first step, a defined number of samples is read from different tracks for every channels. The index of the *outputBuffer* the data is stored to specifies the channel the data is sent to by applying the function *sendToDevice()*. This procedure is repeated until *duration* expires. In the next step, a cross fade is performed for every channel in the same way as described above: the mixed samples are sent to the output device. This procedure is repeated until *fadeTime* expires. After the clean up task and the update of the control variable *current* the process starts repeating the outer while loop.

Static Parallel Mixer

```
while size($\Phi$)>0 do {
    //removes all data from the output buffer
    reset(outputBuffer)

    for current=0:size($\Phi$) {
        //buffer contains information for every channel
        outputBuffer:=outputBuffer+read($\Phi_{current}$)
    }

    //sends the streamed data to the output device
    sendToDevice(outputBuffer)

    //removes all finished streams from $\Phi$
    cleanUp($\Phi$)
}
```

Listing 3.3: The example shows how a Static Parallel Mixer works. It plays back all audio files simultaneously.

The Static Parallel Mixer creates an output signal by mixing all input signals simultaneously. Every audio signal is played back on all channels together with all other signals. Beyond doubt the Static Parallel Mixer represents one of the fastest mixers according to playback time which results in demanding a high degree of concentration of the listeners. Therefore, a long and continued use over hours seems to be impossible due to mental fatigue. Although it may be impossible to recognize single tracks, the mixer can give a quick and rough overview of the audio tracks in a defined area within a cluster structure. The quality of the overview, of course, depends on the underlying structure of data points. In highly heterogeneous areas the Static Parallel Mixer's expressiveness is limited, because the mixed signal contains such a huge number of tracks of different genres that the evaluator is not able to filter the information. In low heterogeneous or nearly homogeneous areas the Static Parallel Mixer can help the listener in quickly detecting tracks which do not fit to the others (outliers). E.g., a selected area contains 15 tracks, 13 classical and two Heavy Metal pieces. For such a structure the Static Parallel Mixer seems to be the best tool for a fast outlier detection. In contrast to that the outlier detection within a structure of 15 tracks which only contains 2 classical but 13 Heavy Metal pieces is nearly impossible. Therefore, the application of the Static Parallel Mixer near clusters could give the listener an appropriate overview of the underlying data within a short time.

Listing's 3.3 pseudo code explains how the Static Parallel Mixer performs the mix-down. All variables and functions used except *reset()* are explained together with listing 3.1. The function *reset()* removes all data from the buffer *outputBuffer*. In the next step a defined number of samples of every signal is stored in the buffer which is passed to the output device next. After applying the clean-up task this procedure is repeated until Φ contains no further signals.

Dynamic Parallel Mixer

The Dynamic Parallel Mixer creates an output signal by mixing all input signals in parallel but with a changing weight. As the Static Dynamic Mixer, every audio signal is played back on all channels together with all other signals. But in contrast to that, one audio signal is played back louder for a defined number of seconds, called *duration*. When the duration time is up the succeeding audio signal is played back louder. The cross-over from one loud signal to the next can be performed soft or hard. The parameter *fadeTime* defines the duration of the fade effect in seconds. A *fadeTime* of nearly 0 seconds forces the mixer to switch to the proximate signal immediately, however often conducted with a disturbing noise. Unlike the Static Parallel Mixer the Dynamic Parallel Mixer highlights tracks in order to play them louder. Therefore, it is easier for the listener to recognize all played back signals while individually presenting an overview of the cluster structure's entire audio content. In order to zoom out the currently louder played track two volume parameters can be specified. The parameter *masterVolume* v_{master} defines the volume level of all signals in percentage which causes a sample value s_{curr} of

$$s_{curr} = s_{orig} * v_{master} \tag{3.24}$$

if s_{orig} is a sample's original value. So a *masterVolume* of 60% sets the volume level of all signals to 60% of their original value. Furthermore, the parameter *volume* specifies the percentage of the volume added to the master volume level of the currently highlighted

```
duration:=3.0
fadeTime:=0.4
masterVolume:=60
volume:=130
current:=0

while size($\Phi$)>0 do {
    //begin: Static Parallel Mixer part
    reset(outputBuffer)

    for i=0:size($\Phi$) {
        //buffer contains information for every channel
        outputBuffer:=outputBuffer+read($\Phi_i$)*masterVolume/100
    }
    //end: Static Parallel Mixer part
    //begin: Dynamic Parallel Mixer part
    if duration has not expired do {
        //add data of currently louder signal
        outputBuffer:=outputBuffer+read($\Phi_{current}$)*(volume/100 - 1)
    }
    else do {
        //add crossover data from louder signal and its succeeding signal
        outputBuffer:=outputBuffer+
            crossover(read($\Phi_{current}$),read($\Phi_{current+1}$))*(volume/100 - 1)
    }
    if duration and fadeTime have expired do {
        current:=nextIndex()
    }
    //end: Dynamic Parallel Mixer part

    //sends the streamed data to the output device
    sendToDevice(outputBuffer)
    //removes all finished streams from $\Phi$
    cleanUp($\Phi$)
}
```

Listing 3.4: Dynamic Parallel Mixer

signal. A *volume* of 120% increases the volume of the currently louder signal by 20% of its original value. If s_{orig} again specifies the original value of the currently highlighted signal s and v_{master} and v_{loud} define the master volume and volume the current sample value s_{curr} is calculated as:

$$\begin{aligned} s_{curr} &= s_{orig} * v_{master} + s_{orig} * (v_{loud} - 1) && (3.25) \\ s_{curr} &= s_{orig} * (v_{master} + v_{loud} - 1) && (3.26) \end{aligned}$$

That means, for the currently louder signal with a *volume* of 120% that it is played with 80% of the original volume level. When increasing the signal's volume level over 180% frequently over-modulation of the signal appears. In that case the *masterVolume* control allows the user to decrease the general volume level which, as a consequence, makes the notion of the currently highlighted signal clearer.

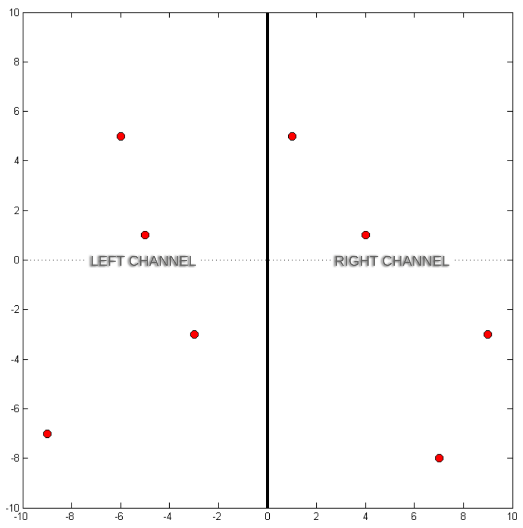
Listing 3.4 presents the function of the Parallel Dynamic Mixer. Besides the parameters *volume* and *masterVolume*, which are explained in the previous paragraph, all variables and functions are discussed either with listing 3.1 or listing 3.3. After a defined number of samples of all signals are stored in the buffer in consideration of the *masterVolume* level the dynamic step starts which in general has the same flow as the Static Sequential Mixer described in section 3.2.3 but additionally takes the *volume* of the currently highlighted signal in account.

Distance Mixer

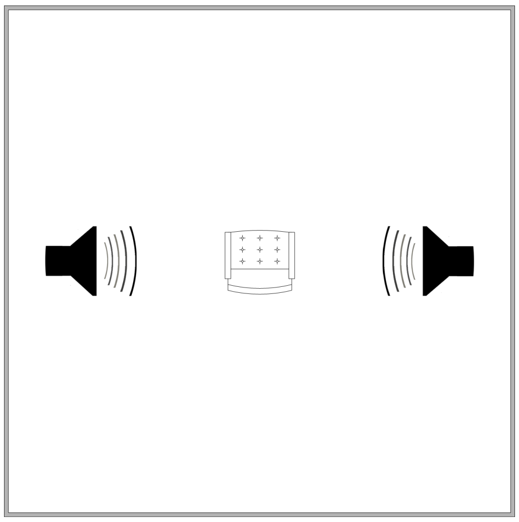
The Distance Mixer creates an output signal on the basis of 2-dimensional, spatial information (see next paragraph and section 2.2.2) of the input signals. That means, every signal is represented by a data point on a plane having an x and y coordinate. The listener himself is located in the origin (0/0) of the coordinate system. The signal's distance to the origin (listener) is expressed by its volume level so that the volume level of a signal decreases with increasing distance to the origin. If δ_{max} denotes the metric measuring of the maximal distance (the distance from the origin to one of the corners) and the origin is located in (0/0) the distance δ of the audio signal i to the listener is defined as:

$$\delta_i = \frac{\sqrt{x_i^2 + y_i^2}}{\delta_{max}} \qquad (3.27)$$

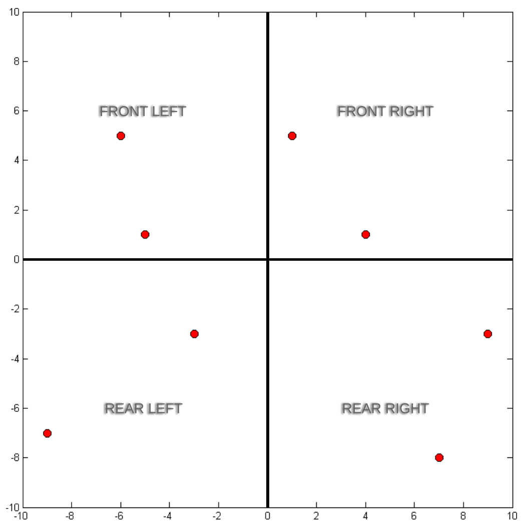
The Distance Mixer supports 2-dimensional playback on two and four channels. Figure 3.4 visualizes eight audio tracks widespread over a plane. For the two channel mode all files located in the negative x range are played back on the left channel and all signals in the positive x range on the right channel. A more specific segmentation is performed in



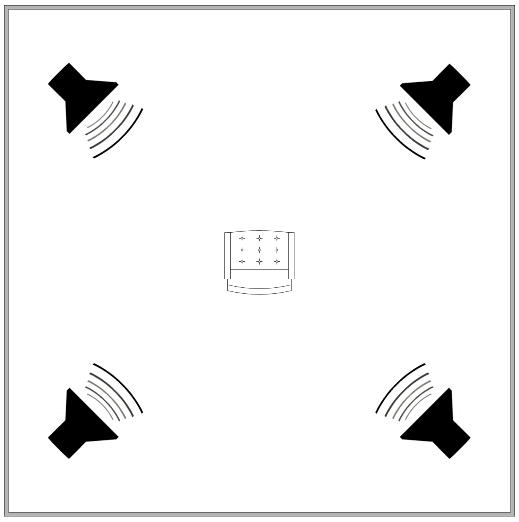
(a) The plot visualizes eight signals distributed on a plane. All signals placed on the negative x side are played back on the left channel, all others on the right channel with a volume level according to their distance to the origin.



(b) displays the positioning of the two speakers for the two channel mode.



(c) The plot visualizes eight signals distributed on a plane. All signals placed on the negative x side are played back in one of the left speakers, all others in one of the right speakers. The signals located in the positive y part are played back in the front, all others in the rear speakers.



(d) displays the positioning of the speakers for the four channel mode.

Figure 3.4: Two and four channel mode of the Distance Mixer.

```
channels:=2

while size($\Phi$)>0 do{
    reset(outputBuffer)

    for current=0:size($\Phi$) do {
        if getXCoordinate($\Phi_{current}$)>0 do
        {
            //buffer contains information for the left channel
            outputBuffer[0]:=outputBuffer[0]+read($\Phi_{current}$)*getDistance(
                $\Phi_{current}$)
        }
        else if getXCoordinate($\Phi_{current}$)<0 do {
            //buffer contains information for the right channel
            outputBuffer[1]:=outputBuffer[1]+read($\Phi_{current}$)*getDistance(
                $\Phi_{current}$)
        }
    }

    //sends the streamed data to the output device
    sendToDevice(outputBuffer)

    //removes all finished streams from $\Phi$
    cleanUp($\Phi$)
}
```

Listing 3.5: Distance Mixer example

the four channel mode in which the positive and negative x range in each case is split into two areas. All signals located in the positive y range are sent to the front channel (left or right) and all in the negative y range are sent to the rear channel. Figure 3.4 illustrates the positioning of the front and rear speakers around the listener.

Based on coordinate information every audio signal is played back on a specific channel together with all other signals which appear in the same area having a defined volume level. According to the example of figure 3.4 the listener hears four signals per channel in the two channel mode and two signals on every channel in the four channel mode. Because the Distance Mixer plays back all signals in parallel it is ranked to the Parallel Mixers segment. As for all other Parallel Mixers the playback time is very short which also results in the need of a high degree of concentration of the listener. The Distance Mixer also presents an overview of the underlying data points but the most detailed because of taking the position of the signal within the area and its distance to the listener into account. That allows the evaluator to recognize if a possible outlier is located in the middle or the boundary of the area. The problem which occurs in very densely populated areas is that tracks having a short distance could drawn all other tracks of its channel out.

Listing 3.5 describes the Distance Mixer in the two channel mode. It works in the same way as the Static Dynamic Mixer which is discussed in section 3.2.3 but is extended by the functions *getXCoordinate()* which returns the x-coordinate of passed signal. Furthermore, the variable *outputBuffer* represents an array of buffers which is organized in a way that all data stored in index 0 are sent to the left and all of index 1 are sent to the right channel by applying the function *sendToDevice()*.

Spatial Information

The Distance Mixer which is discussed in the paragraph above uses 2-dimensional, spatial information in order to present the listener a very detailed overview of the underlying data points of a cluster structure. According to the run-time mode the application is executed in the source of the spatial information differs.

If cLynx extends the PlaySOM which is discussed in section 2.2 as Plug-In (see Plug-In Mode in section 3.3.4) the spatial information is provided by the underlying system. As explained in section 2.2.2 the exact position of an input is calculated under consideration of its at least three best matching units on a SOM. The resulting spatial information and the corresponding audio signals are passed to cLynx through the interface which is discussed in section 3.3.4 in order to perform an evaluation using the Distance Mixer.

In the Stand-Alone Mode (see section 3.3.1) the 2-dimensional information which specifies the exact position of pieces in the clustering structure must be defined by the user due the absence of an underlying application which could pass that kind of information. For that reason the directory of the audio files additionally must contain an obligatory file named *sample.coordinates* which specifies the x and y coordinates of all tracks in the directory. The coordinates file is built up like a simple property file using the format of a Key-Value list like *key=value*. Except one predefined key all other keys are specified through the name of the audio tracks. The specification of the x and y coordinates of two of the tracks evaluated in cLynx of figure 3.5 looks like:

```
(track)=(x-coordinate)[ ,;](y-coordinate)
myTrack1.wav=-1,0
myTrack2.mp3=8,0
```

If a sound file's coordinates are not specified the track is not considered by the Distance Mixer which results in total silence when no coordinates file is specified. Additionally to the spatial information of the tracks the width of the unit must be defined through the key *width* used to calculate δ_{max} which is part of equation 3.27. The parameter *width* defines the horizontal length of the unit which additionally is the height as cLynx always supposes the underlying plane as a square. When representing the data of figure 3.4 *width* can be initialized with 20 for instance. Furthermore, *width* could also have the value 18 because all points lie in a coordinate system with an x-axis of [-9,9] and y-axis of [-9,9]. Acoustically the *width* of 20 differs to the *width* of 18 in the volume level of the tracks because it is part of the calculation of the maximal distance δ_{max} to the listener. According to the equation 3.27 the distance of a signal δ_i decreases with increasing δ_{max} which furthermore means that the signal's volume level increases with increasing δ_{max}.

3.2.4 Extending cLynx

In order to support further research cLynx is implemented in a way that the extension of new components is easily enabled. Whenever a new analysis, normalization or mixer component has been implemented based on the information given in the following paragraphs, the component must be made available to cLynx which differs for several Run-Time Modes.

To make a new component available in the Console Mode the class *Constant* has to be modified by enabling it to identify the Integer representation of the new component which is passed through the *settings.properties* file. To ensure this the getter-methods of the specified component have to be extended.

The modulations in order to make new components visible in the GUI Mode are much more complex than the extensions required by the Console Mode. First of all, a new instance of *JDefPanel* must be defined which graphically represents the new component and contains all its parameters. It must be added to the corresponding container class *JSelectionPanel* which manages the switching through the several *JDefPanel* instances. All parameters of the new *JDefPanel* must be defined through the *setInputComponent()* method which enables them being editable in run-time when setting the flag *chMonitor* to *true*.

Analysis Component

In order to implement an additional analyzer, the programmer's interest should be focused on the *audio.statistic* package. Every new analysis component must extend the abstract class *AbstractAnalyser* and has to define its *ID* variable in the constructor. In the second step the calculation of the values has to be implemented in the class *AnalyserTool*. It is important that for every analyzer the maximum and minimum values are calculated, otherwise it will result in a Java NullPointerException.

Normalization Component

When extending the normalization component, the developer should be focused on the *audio.norm* package. Every new normalization process must implement the Java interface *INormalization*. An abstract Compressor is implemented in the abstract class *AbstractCompressor* which should be extended when implementing another Compressor instance.

Mixer Component

The most complex extension is represented by the implementation of an additional mixer component. In this case the package *audio.mux* contains all relevant classes. Every new mixer is represented as a stream and therefore must extend the abstract class *AudioStream*. In the constructor of every new mixer process the variables

- *max_parallel_tracks*
- *input_to_output_channel*
- *_PROVIDER*
- *_FILES*
- *_SAMPLE_LIST*

have to be defined. For further information of the several data structures read the relevant passages of cLynx's Java API.

3.3 Run-Time Modes

The cLynx tool can be executed in several different run-time Modes which will be discussed in detail in the following paragraphs. The application flow within cLynx will not highly differ between the different modes. The two main modes are represented by the:

- Stand-Alone Mode
- Plug-In Mode

While the Plug-In Mode is only executable using a graphical interface the Stand-Alone Mode additionally allows starting the application through a command line in the console which furthermore makes cLynx distinguishable between:

- Console Mode
- GUI Mode

The main difference between all modes is represented by the modality of parameter passing. While the GUI Mode allows the definition of the parameters through several dialogs the Console Mode needs a property file which contains all required parameters. In the first

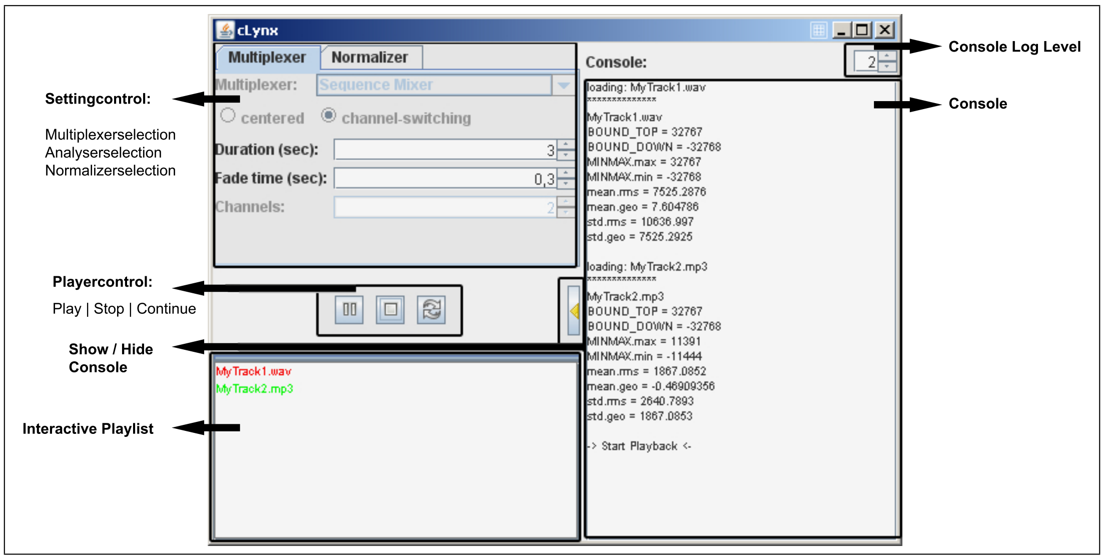
Parameter ID	Must	Type	Default Value
audio_dir	y	Url	n/a
ext	y	String	n/a
repeat	n	Boolean	false
fadeIn	n	Double	1.0
mux	n	Integer	0
duration	n	Double	3.0
fadeTime	n	Double	0.3
targetChannels	n	Integer	2
volume	n	Double	0.8
mastervolume	n	Double	0.6
distanceChannels	n	Integer	2
normalizer	n	Integer	0
compressionrate	n	String	4:1
attack	n	Integer	50
release	n	Integer	50
analysis	n	Double	0
threadLoading	n	Boolean	false
threadNO	n	Integer	2

Table 3.1: All keys of the setting file which passes all required parameters to cLynx in the Stand-Alone Console Mode.

part of this section both Stand-Alone Modes are described followed by an explanation of the Plug-In Mode. In the last part developers are guided through the integration of the cLynx plug-in into their applications.

3.3.1 Stand-Alone Mode

The Stand-Alone Mode allows the usage of cLynx without an application which passes the required parameters to the tool. Therefore, the user is responsible to pass all required parameters to the tool which makes this mode especially suitable for test runs and predefined evaluation scenarios. When executing cLynx in the Stand-Alone Mode a *coordinates file* has to be created which is documented in section 3.2.3, otherwise the Distance Mixer can not be used.



```
loading: MyTrack1.wav
**************
MyTrack1.wav
BOUND_TOP = 32767
BOUND_DOWN = -32768
MINMAX.max = 32767
MINMAX.min = -32768
mean.rms = 7525.2876
mean.geo = 7.604786
std.rms = 10636.997
std.geo = 7525.2925

loading: MyTrack2.mp3
**************
MyTrack2.mp3
BOUND_TOP = 32767
BOUND_DOWN = -32768
MINMAX.max = 11391
MINMAX.min = -11444
mean.rms = 1867.0852
mean.geo = -0.46909356
std.rms = 2640.7893
std.geo = 1867.0853

-> Start Playback <-
```

Figure 3.5: cLynx's graphical user interface.

3.3.2 Console Mode

The Console Mode allows the user to execute the application through a command line in the console without needing a GUI. Before starting cLynx a property file must be created, which contains all required parameters. Table 3.1 illustrates all required parameters of the settings file and their default values. The range of valid values for each key is equal to the possible values listed in table A.1. Apart from the parameters *audio_dir* and *ext* all others do not have to be specified as default values are defined for them. To execute the tool in this mode use the command:

java -Xmx1500M -cp [CLASSPATH] as.clynx.CLynx -s [PATH TO SETTINGS FILE]

3.3.3 GUI Mode

The GUI Mode allows the user to execute cLynx with a graphical user interface by the command:

java -Xmx1500M -cp [CLASSPATH] as.clynx.CLynx

The GUI allows the definition of all parameters directly within several dialogs. The cLynx's graphical user interface which is depicted in figure 3.5 can be divided into four main parts. All parameters for the mixer, analyzer and normalizer can be specified in the *Settingcontrol*. In contrast to the Console Mode several parameters can be changed in run-time by the user. By using the player buttons in the *Playercontrol* the playback can be started, paused and stopped. When pushing the play button an open dialog appears which asks the user to specify the path to the audio files. The selection of the repeat button enables or disables endless playback of the tracks. The *Interactive Playlist* visualizes all tracks which are evaluated and highlights the currently active track in green and already finished tracks in red. Some mixer, e.g. both Sequence Mixer and the Dynamic Parallel Mixer, allow direct interaction through mouse selection in the play list. A double click event during the playback with selected Static Sequential Mixer forces the player to switch to the selected track. The Dynamic Sequential Mixer is forced to play the selected track on the front left channel and the Dynamic Parallel Mixer plays the selected track louder.

The right side of the graphical user interface is represented by the *Console*. The *Console* can be hidden and shown again by pushing the *Show/Hide Console* button in the center. The *Console Log Level* defines which and how much information should be displayed in the console. When selecting a log level of 2 all messages with an equal or higher priority than 2 are displayed in the console field, all lower than 2 do not pass the information level filter.

3.3.4 Plug-In Mode

The Plug-In Mode allows developers to integrate cLynx in their application. In contrast to the Stand-Alone Mode only the graphical representation is available in the Plug-In Mode. The application flow does not differ from the Stand-Alone GUI Mode except for the passing the parameters. The spatial information which is explained in detail in section 3.2.3 and the corresponding tracks are provided by the underlying system and passed to cLynx through communication interfaces which are accessible in cLynx's package *interfaces*. The next paragraphs document how cLynx can be integrated in other applications.

Integration of cLynx

The cLynx plug-in can easily be added to a Java Swing[1] application as it extends the *JPanel* class of Java's Swing package. Listing 3.6 demonstrates how cLynx has to be integrated in other applications. The two most important commands are the explicit announcement that cLynx is in the Plug-In Mode which is performed through the line *CLynx.PLUG_IN = true* and the passing of an interface which manages the communication between the extended system and the plug-in. This is done by applying the function *setCLynxConnector()*.

The communication between the underlying system and cLynx is managed by cLynx's communication interface *CLynxConnector*. It sends the spatial information (see section 3.2.3) and its corresponding audio tracks to cLynx. As mentioned above, the interface is set by the function *setCLynxConnector()*.

[1]Swing is part of Sun Microsystems Java Foundation Classes (JFC), which covers a group of features for building graphical user interfaces (GUIs). JFC additionally encompass classes for adding 2D drawing functionality and interactivity to Java applications. [Horton, 2000]

```
GridBagConstraints constraints = new GridBagConstraints();
constraints.fill = GridBagConstraints.BOTH;
constraints.weightx = 0.1;
constraints.weighty = 0.1;

// Enables Plug-In Mode
CLynx.PLUG_IN = true;

//creates a new GUI
CLynx tool = new CLynx();
//sets the communication interface
tool.setCLynxConnector(this);
getContentPane().add(tool, constraints);
```

Listing 3.6: This code simply instructs how to integrate cLynx into an external application.

3.4 Summary

In this chapter the cLynx tool was discussed which aims at supporting users in acoustically evaluating music similarity measures. It offers diverse features for presenting a meaningful signal to the user. Several Mixer components perform different mix-down types, primarily based on parallel and sequential playback. While simple simultaneous playback is extended by highlighting single songs, sequential playback can be performed by playing a track on all channels or by switching it through the channels. A special mixer instance extends parallel playback by generating an acoustic representation of the signals' spatial appearance on a 2-dimensional lattice. According to the run-time mode, the required coordinates are provided by the user or an underlying system, which is extended by cLynx. The plug-in mode of cLynx easily can extend applications, what already is realized with the PlaySOM application. The user study presented in the next chapter is based on the interplay of both systems.

Chapter 4

Evaluation Results

4.1 Test-Set 58

4.2 Analysis Tuning 58

4.2.1 Procedure 59

4.2.2 Random Selection Approach 60

4.2.3 Uniform Selection Approach 63

4.2.4 Summary 64

4.3 User Study 64

4.3.1 Parameter Analysis 64

4.3.2 Subjective Evaluation 68

4.4 Summary 78

This chapter presents the results of two separate evaluations. The first study concerns the cLynx's most time-expensive part, the Analysis Component. In order to tune the performance of this component two faster approaches for accurate description value extraction are analyzed. Both methods calculate these values by only processing the music signals partially.

In the second part of this chapter an evaluation is discussed which analyses the applicability of cLynx for music similarity judgment and outlier detection. Additionally, the study is focused on the human cognition of music and subjective music similarity judgment. Furthermore, the evaluation tries to detect connections between subjective interpretation and individual preferences and the subjects' perception of music similarity and susceptibility to outlier detection.

Genre	**#Songs**
Classic	640
Electronic	229
Jazz and Blues	52
Metal and Punk	90
Rock and Pop	203
World	244

Table 4.1: Organization of the ISMIRgenre test set

4.1 Test-Set

All evaluations were performed on the ISMIRgenre collection [Lidy and Rauber, 2005] from the Audio Description Contest on Genre Classification of the International Conference on Music Information Retrieval 2004 which contains 1458 audio files. All songs in the collection are organized in six genres each containing a different number of tracks which are presented in table 4.1. The compiled test set does not contain full tracks but 30 second long pieces available in 128 kbps, 44 kHz, stereo MP3 format. Furthermore, the test set was converted to 16 Bit, 44 kHz, PCM signed, stereo WAV format for the analysis tuning evaluation in section 4.2.

4.2 Analysis Tuning

As already discussed in section 3.2.1 the analysis component represents a fundamental step for presenting the evaluator a meaningful output signal. It is responsible for computation of the so called description value sets of all audio signals passed which build the basis for the calculation of the global upper and lower boundary and contain the following values:

- Minimal Sample Value
- Maximal Sample Value
- Arithmetic Mean Value
- Root Mean Square Value
- Standard Deviation Value

The calculation of a set of description values is performed by processing an audio file entirely which on the one hand provides accurate results but on the other hand is cLynx's most time-expensive task. Therefore, two other approaches are evaluated in order to lower the computation time of the analysis. Both processes only use parts of the underlying audio data extracted in a random or uniform way which is explained in the following paragraphs.

4.2.1 Procedure

In order to accelerate the time-expensive analysis task the calculation of a set of description values is performed by processing the underlying audio data only partially. If therefore the computation of the description value set of the signal i is based on 50% of the signal's data the resulted set is defined as $\Theta_{i,50}$. In order to have a reference value the original description value set additionally is calculated by using the conventional method which, of course, uses 100% of the data. So the original description value set of the audio signal i is defined as:

$$\Theta_i = \Theta_{i,100} \tag{4.1}$$

The differences δ between all single values of Θ_i and $\Theta_{i,50}$ build the set $\Delta_{i,50}$ and can be calculated by a simple subtraction of the partial values from the original values for every file i:

$$\delta_{i,max} = \Theta_{i,max} - \Theta_{i,50,max} \tag{4.2}$$
$$\delta_{i,min} = \Theta_{i,min} - \Theta_{i,50,min} \tag{4.3}$$
$$\delta_{i,\mu} = \Theta_{i,\mu} - \Theta_{i,50,\mu} \tag{4.4}$$
$$\delta_{i,rms} = \Theta_{i,rms} - \Theta_{i,50,rms} \tag{4.5}$$
$$\delta_{i,\sigma} = \Theta_{i,\sigma} - \Theta_{i,50,\sigma} \tag{4.6}$$

So Δ_i describes the distance between the original set and the set calculated by only using parts of the data. In order to make the resulting values more transparent the distances are smoothed and normalized. Smoothing the data is done by averaging all resulting sets $\Delta_{i,50}$ of a common genre which builds the set Δ_{50}. To do that 15 audio files of each genre are selected randomly. The normalization of the results is based on a value which represents the greatest possible oscillation for the audio signals. Because the whole test was performed on 16 Bit PCM SIGNED, stereo tracks (see section 4.1) the largest sample value is represented by the value 2^{15}.

High distances within a set Δ point out that the description value set calculated using only partial source data does not give an accurate description of the underlying audio signal's structure. Based on highly deviating values the calculation of the global upper and lower boundaries which are explained in section 3.2.1 does not give accurate results which causes inefficient normalization. As a consequence, loud signals are still too loud after the normalization process and therefore could drawn quiet tracks out.

	Classic	Metal & Punk
value	**arithmetic mean**	**arithmetic mean**
δ_{max}	0.124	0.004
δ_{min}	0.046	0.023
δ_{μ}	3.72E-05	6.01E-05
$\delta_{\mu rms}$	0.008	0.004
δ_{σ}	0.008	0.004

Table 4.2: Averaged, normalized differences of Δ_{50} and a π of 10 for the genres Classic and Metal.

4.2.2 Random Selection Approach

Instead of processing an audio file entirely the random approach deals with the analysis of only a predefined number of randomly selected parts of an audio file. The degree of partitioning is defined through the partition coefficient π. A π of 10 specifies a partitioning of an audio signal into ten consecutive parts with equal length in order to aggregate 10% of the underlying audio information for every part. The calculation of the description value set is performed on the basis of randomly chosen 50% of those parts which means taking 5 random samples from the partitioned audio signal. So the resulting description value set $\Theta_{i,50}$ is based on randomly chosen 50% of the signal's data. As described in section 4.2.1 all $\Theta_{i,50}$ of 15 randomly selected audio files of a common genre build the basis for the calculation of the distance set Δ_{50}.

Table 4.2 presents the normalized differences of the Δ_{50} set of Classical music using a partition coefficient of 10. Except δ_{max} and δ_{min} all other values are in an acceptable range of $< 1\%$. Especially the difference of the arithmetic mean δ_{μ} is near 0. In contrast to that a divergence of more than 12% of the description value δ_{max} is very high. It means that the averaged divergence between the description values $\Theta_{i,50,max}$ and $\Theta_{i,max}$ of a Classical file i amounts 12% of the signal's maximal possible oscillation, as shown in figure 4.1. Even when examining Δ_{70} using the same partition coefficient the maximal difference decreases but again is greater than 12%. Therefore, both resulting description value sets, Θ_{50} and Θ_{70}, are not representative for the underlying signal due too high divergence from the original values. Based on such differing values the calculation of the global upper and lower boundaries does not give accurate results, causing inappropriate inefficient normalization. As a consequence, loud signals are still too loud after the normalization process and therefore could drawn quiet tracks out. An indicator for the high value of δ_{max} and δ_{min} can be found in the dynamic structure of the signals which is discussed in detail in section 2.3.1. Because Classical tracks are such high dynamical pieces the detection of

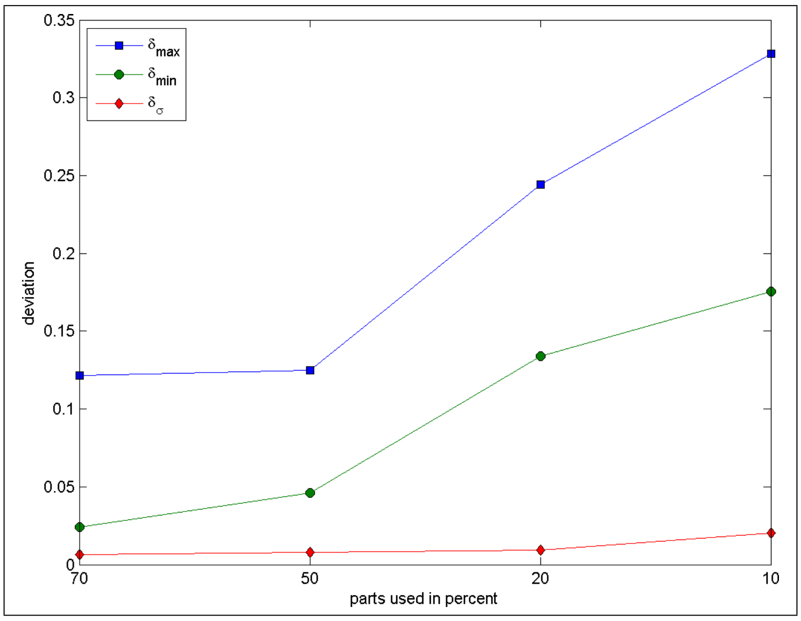
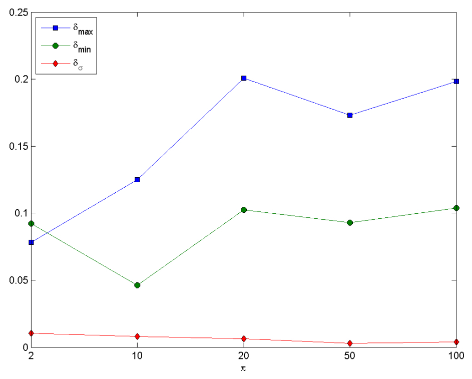


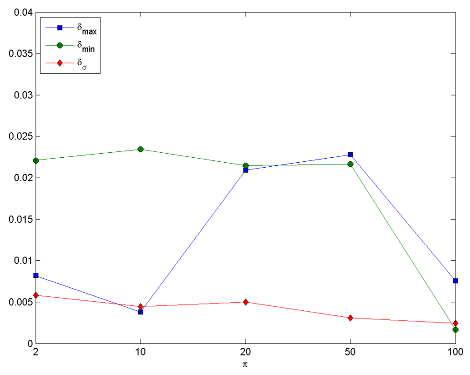
Figure 4.1: Trend of δ_{max}, δ_{min} and δ_σ for the sets Δ_{70} to Δ_{10} and a π of 10 for the genre Classic.

its greatest oscillation is difficult. In contrast to Classical pieces Heavy Metal and Punk tracks are considered to low dynamical signals because most of the Heavy Metal tracks start with high volume and continue with that volume level until the end. Therefore, the same test is repeated using 15 randomly selected tracks of the genre Metal and Punk which again are divided into equal-sized parts by using a partition coefficient of 10. The results when taking 5 random samples of each partitioned signal are presented in table 4.2. The values of δ_{max} and δ_{min} are much smaller compared to the results of the Classical pieces because tracks of this genre reach the maximal oscillation more often. Figure 4.1 illustrates the trend of the differences δ_{max}, δ_{min} and δ_{sigma} of the sets Δ_{70}, Δ_{50}, Δ_{20} and Δ_{10} for the genre Classic when using a fixed π of 10. It confirms the general assumption that all differences of a set Δ increase with decreasing number of sample takes, which should be valid for all genres.

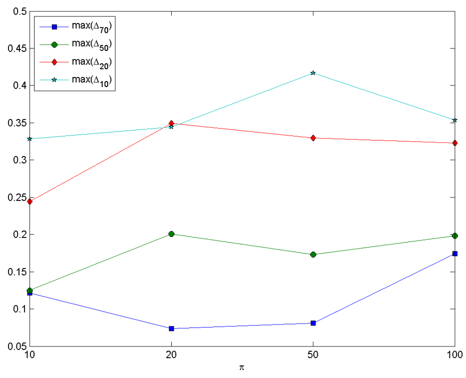
In order to demonstrate the statistical processing, an exemplary audio signal was divided into consecutive parts with equal length in order to aggregate 10% of the underlying audio information for every part. This rather coarse partitioning of the audio signals was deliberately chosen to clearly describe the estimation of desired statistical moments although it definitely causes a strong variation between the respective estimates of every part. Therefore, the degree of partitioning is increased, which as a consequence, decreases the audio information aggregated by each part. That means, the set Δ_{50} was calculated by taking more but smaller samples in order to increase the distribution of the samples which should result in more accurate description values. Figure 4.2(a) and 4.2(b) depict the trends δ_{min} and δ_{max} of the set Δ_{50} according to varying partition coefficients for



(a) $\delta_{max}, \delta_{min}$ and δ_σ of Δ_{50} for varying π of the genre Classic



(b) $\delta_{max}, \delta_{min}$ and δ_σ of Δ_{50} for varying π of the genre Metal and Punk



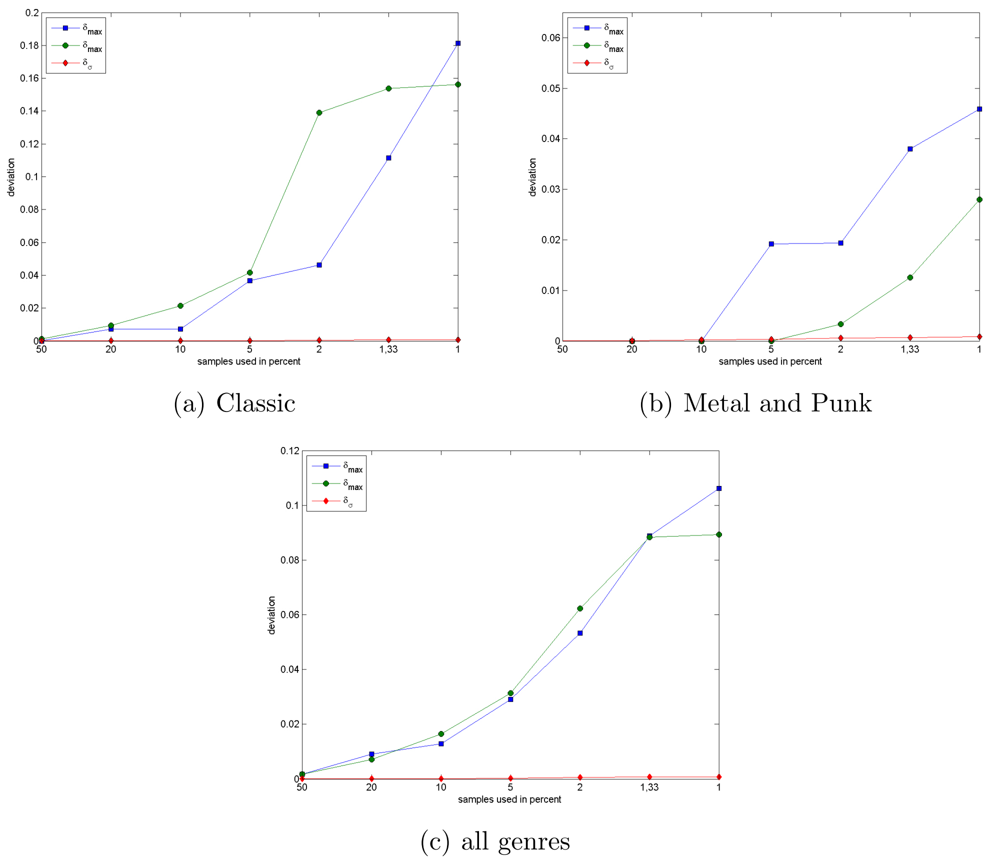
(c) All averaged maximal differences of the sets Δ_{70}, Δ_{50}, Δ_{30} and Δ_{10} of all genres using different aggregate coefficients.

Figure 4.2: Several differences of various Δ sets for varying partitioning.

the genres Classic and Metal. Contrary to the assumption the best approximation for the genre Classic occurs for a partition coefficient of $\pi = 2$ which represents the case of splitting the signal into two parts and either taking the partition which contains the beginning or the end of a piece of music. The trends of δ_{min} and δ_σ for the genre Metal and Punk for varying π permanently lay under 2.5% and is nearly constant for a π of 2 to 50. Both genres represent the maximal and minimal extreme of all sets Δ from all genres over all degrees of partitioning. Figure 4.2(c) depicts the trend of all averaged maximal differences of the sets Δ_{70}, Δ_{50}, Δ_{20} and Δ_{10} of all genres defined as:

$$\max(\Delta_x) = \max(\delta_{max}, \delta_{min}, \delta_\sigma, \delta_\mu, \delta_{rms}) \qquad \delta \in \Delta_x \tag{4.7}$$

Again, the trends disagree with the assumption that the quality of Θ raises with increasing π but it illustrates that the difference between the original and approximated description



(a) Classic (b) Metal and Punk

(c) all genres

Figure 4.3: $\delta_{max}, \delta_{min}$ and δ_σ for various genres on uniformly distributed sample takes

values decreases with increasing number of sample takes. While the divergence of the maximal value of the set Δ_{10} constantly is grater than 30%, the maximal values of the sets Δ_{50} and Δ_{70} nearly never exceed 20%.

4.2.3 Uniform Selection Approach

Additionally to the random test the experiment is extended by the calculation of the description values based on uniformly distributed sample takes. So the set Θ_i of a randomly chosen audio signal i is computed based on a set S of audio sample values created by uniform takes from the source signal. If therefore S aggregates 50% of the data it contains every 2^{nd} sample value of the underlying audio signal and if S contains 20% of the original source it considers every 5^{th} sample etc. So of each arbitrarily sized part of the signal is taken the same number of samples. As a consequence, the quality of Θ should be much better compared to the results presented for the random approach. Figure 4.3(a) illustrates the trend of δ_{max} and δ_{min} for decreasing aggregation of S from 50% to 1% of the source data. Even when using 10% of the underlying data the maximal difference is smaller than 2.2% of the signal's maximal oscillation which is a great improvement compared to the results of the random approach shown in figure 4.1. Furthermore, the results of the genre Metal and Punk are depicted in figure 4.3(b). There is no deviation from the original values for Δ_{50} to Δ_{10} and a difference of 4.6% when using only 1% of the underlying data. Although figures 4.2(c) and 4.3(c) are not completely comparable due to different partition sizes, they illustrate that the calculation based on uniform selection of 1% of all samples creates description values of higher quality than the computation based on randomly chosen 50% of the samples.

4.2.4 Summary

The evaluation presented two different approaches for optimizing the analysis process. The results pointed out that the random approach does not produce accurate description values for the underlying song. Interestingly, the increase of the distribution of samples taken by rising π did not improve the quality of the resulting description set Θ. By contrast, the uniform approach works significantly better. As shown in figure 4.3(c) the averaged deviations over all genres of Θ_{50} (0.2%) and Θ_{20} (0.9%) to the original description values are minimal.

4.3 User Study

Because of raising demand of innovative structuring approaches for huge music repositories, science steadily is challenged to find novel methods for organizing data. Most approaches perform an organization by specifying a special similarity measurement, based on the human intuitive process of structuring. The key problem arising with the definition of similarity is the subjective human interpretation of music similarity. Human cognition of music inherently is biased by subjective interpretation and reasoning which usually is based on knowledge and conventions of the real world. Therefore, it makes sense measuring the quality of music similarity in a subjective way based on listening tests. Several user studies have been applied to this problem which are reviewed in section 2.4.2.

In this section a user study is presented which is focused on human cognition of music and subjective music similarity judgment. The underlying organization method applied to a music data set is based on the Self-Organizing Map approach. The listening test was performed using the cLynx tool which extends the application PlaySOM (see section 2.2) as plug-in.

4.3.1 Parameter Analysis

cLynx, which is discussed in detail in chapter 3, offers several features for analyzing, normalizing and mixing audio data. Several of these functions enable the user to define parameters to directly influence the character of the resulting mix-down. In order to make

ID	% louder
DPM_1	50%
DPM_2	100%
DPM_3	150%
DPM_4	200%

Table 4.3: Several configurations of the Dynamic Sequential Mixer. The ratio item specifies the level the current song is played louder than the background noise.

the user study in section 4.3.2 more transparent a pre-study was performed to discover the optimal configuration of required parameters. The parameters evaluated by this study are:

- *duration*: The *duration* specifies the time a song can be perceived by the listener. This parameter is required for both Sequential Mixers and the Dynamic Parallel Mixer.
- *volume* and *master-volume*: The *volume* and *master-volume* parameters are used to highlight songs when using the Dynamic Parallel Mixer.

The pre-study was performed on four subjects using cLynx. On a music map based on the music data set discussed in section 4.1 each of the subjects was acoustically presented four regions of the PlaySOM application. All specified areas contained at least six tracks and although covering outliers (e.g. songs not fitting to the rest of songs in the area) each region could be clearly assigned to one genre. While two regions covered Classical songs the others contained Rock and Electronic tracks. The areas assigned to Classic could be clearly distinguished between singing based and instrumental oriented music. The playback was performed by the cLynx plug-in using different parameter configurations. Six times every region was presented to each subject using constant *duration*, but changing Mixer components. So the presentation of one area used both Sequential Mixers and several Dynamic Parallel Mixers using different volume levels. Table 4.3 illustrates the various configurations of the Dynamic Parallel Mixer component used for every region related audio presentation. The various Dynamic Parallel Mixer settings differ in their volume level a highlighted song is played louder than the background noise. E.g., the highlight effect of DPM_1 plays highlighted tracks 50% louder. After every presentation each subject was interviewed using the questions listed in table B.1 in appendix A.

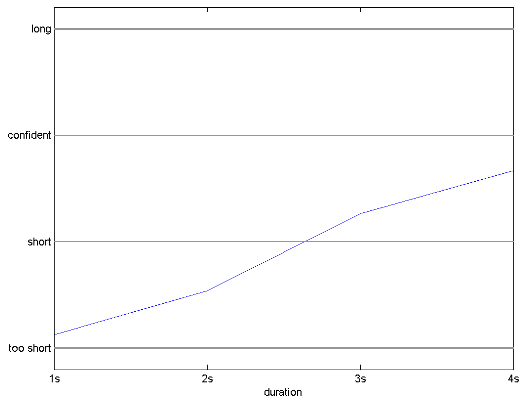
Duration Parameter

The ideal setting of the duration parameter was evaluated by presenting every subject four regions with differing duration times. The duration parameter was fixed at $1, 2, 3$

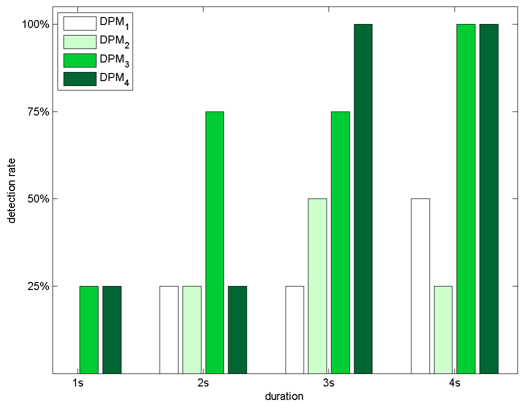
Mixer	Duration	Genre Identification	Error #Songs	Confidence Level
SSM	1	100%	4.25	8.25
DSM	1	75%	3.75	7.75
DPM_1	1	75%	4	6.5
DPM_2	1	50%	4.5	6.5
DPM_3	1	50%	3.75	6.75
DPM_4	1	75%	3.5	7.5
SSM	2	100%	1.5	8.25
DSM	2	100%	1.25	8.25
DPM_1	2	100%	3.25	7
DPM_2	2	100%	3.5	7.5
DPM_3	2	100%	3.5	7.75
DPM_4	2	100%	3	8.25
SSM	3	100%	1.75	8.5
DSM	3	100%	2.5	8.5
DPM_1	3	75%	3.75	6.5
DPM_2	3	100%	3.5	7.5
DPM_3	3	75%	3.75	8
DPM_4	3	75%	2.75	8.25
SSM	4	100%	2	8.5
DSM	4	100%	3	8.5
DPM_1	4	100%	2.25	7
DPM_2	4	100%	2.25	7.5
DPM_3	4	100%	2.5	7.75
DPM_4	4	100%	2.25	8.25

Table 4.4: Correct genre identification and difference of the detected number of songs to the real number of songs and the subject's confidence level according to the *duration* parameter and Mixer component

and 4 seconds having constant value for all region related listening situations. After every listening situation each subject was asked whether the duration time had comfortable length or not. As illustrated in figure 4.4(a) the subjects sensed durations smaller than 3 seconds as short and felt nearly comfortable with a duration value of 4 seconds. Additionally, this trend was confirmed by the subject's bearing. While seeming to be overburdened when being confronted with a duration time smaller than 3 seconds, the subjects' behavior expressed that they worked with full capacity without being overrun when the duration



(a) Subject's feeling for *duration* parameter.



(b) Subject's ability to identify the highlighting effect of the Dynamic Parallel Mixer for varying *duration*.

Figure 4.4: Participant's feeling of duration and identification of DPM's highlighting effect.

was fixed at 4 seconds. Besides the subjects consistently felt more confident with their answers when using one of the Sequential Mixers, which is expressed by the confidence level[1] in table 4.4, a better identification of the highlighting effect of the Dynamic Parallel Mixers emerged for DPM_3 and DPM_4. This trend became more apparent with increasing duration. Nearly all subjects could recognize the highlighting effect of DPM_3 and DPM_4 when using a duration of length 3 or 4 seconds. The trend of all Dynamic Parallel Mixer configurations regarding changing duration is depicted in figure 4.4(b). Remarkably, although the subjects sensed a duration of 2 seconds as short, they were able to detect the correct genre and to estimate the number of perceived songs better than for a duration of 3 seconds as shown in table 4.4. Nevertheless, they felt more confident with their answers when the duration was fixed at 3 seconds. However, except this fact the correct identification of the region's assigned genre and the estimation of the number of perceived songs yield better results with longer duration.

Volume Level

As already mentioned in the paragraph above the detection rate for the various Dynamic Parallel Mixer configurations listed in table 4.3 raised in parallel to the increase of the duration of playback of individual tracks. Figure 4.4(b) illustrates that only few subjects recognized the highlighting effect of the Dynamic Parallel Mixer for a duration lower than 3 seconds. Although most subjects thought of perceiving several songs nearly in the same volume level the mixer configurations DPM_3 and DPM_4 were better identified. This

[1]All participants were ask to rate how confident they felt with their answers. A score of 1 denotes unconfident and 10 means very confident.

trend is confirmed by the use of a duration of 3 and 4 seconds when nearly all subjects recognized the highlighting effect of the configurations DPM_3 and DPM_4 as depicted in figure4.4(b). Furthermore, the users felt more confident with their answers when improving the highlighting ratio of the Dynamic Parallel Mixer which is illustrated in table 4.4.

4.3.2 Subjective Evaluation

As described in this section's introduction subjective music similarity judgment is a very important and sophisticated task, because of the differing human cognition of music and interpretation of acoustic similarity. Therefore, the evaluation of music organization approaches based on music similarity measurements typically is performed by subjective listening tests. Evalutron [Gruzd et al., 2007], an evaluation tool developed by Stephen Downie's IMIRSEL lab for MIREX 2006 and which is discussed in section 2.4.1, supports subjects in analyzing and judging music similarity. A novel approach which is applied to assist subjects in subjective music similarity analysis and judgment is the cLynx tool, which is presented in chapter 3 of this thesis.

The user study, which is discussed in the following paragraphs, was performed based on the cLynx tool. The key goal of the study is to analyze the human cognition of music and the subjective interpretation of heard audio data regarding different listening conditions simulated by the various mixer components of cLynx. The human's ability of filtering music (outlier detection) and judging music similarity and the subjective level of listening enjoyment according to sequential and parallel music playback were evaluated in order to find preferred mixer types for specified cluster regions visualized by the PlaySOM application.

Setup

13 human subjects participated in the evaluation. The subjects coming from varying professional guilds were equipped with differing musical pre-conditions and previous technical knowledge. 9 participants enjoyed instrumental education in their childhood and 2 of them play music with an instrument this very day occasionally. The test was performed using the PlaySOM application (see section 2.2) which organized the ISMIRgenre music collection discussed in section 4.1. The playback was performed by cLynx plugged into PlaySOM.

Prior to every evaluation process each subject was shortly introduced into the area of Music Similarity Systems on the basis of the PlaySOM application. After this explanation every subject was aware of what a Music Similarity System is and how a 2-dimensional representation of music libraries could look like without knowing technical concepts or possible performance values. In order to present the subjects a better audio quality all

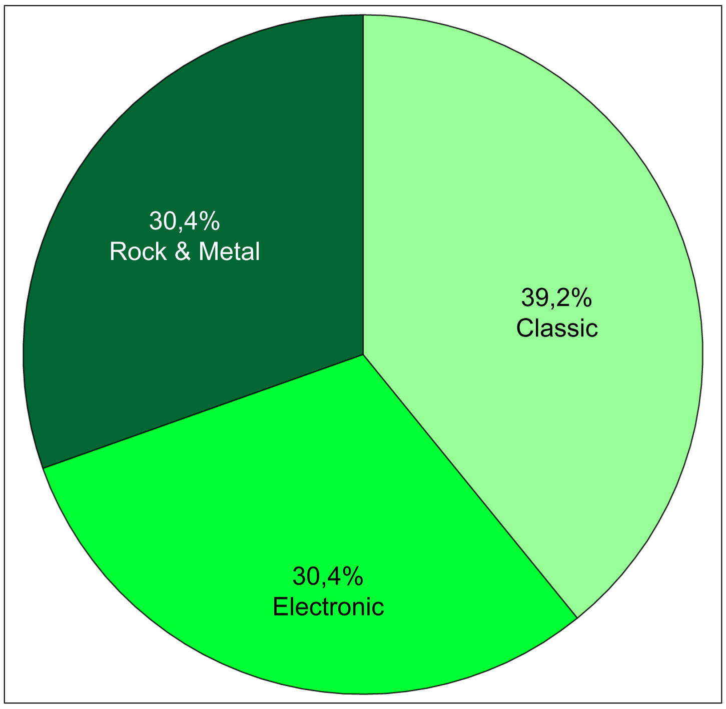


Figure 4.5: Genre based segmentation of all listening situations.

listening tests were performed using an ear-phone. Thus, the results were not actively influenced by disturbing noise of the environment. The ear-phone contained a volume control which enabled every subject to specify the master volume by her/himself.

Every participant was acoustically presented various regions of the PlaySOM application using the cLynx plug-in with varying configurations. Each presentation, called listening situation, used special mixer configurations which did not change between the subjects. So every listening situation was equal for all subjects. cLynx's defined mixer settings were derived from the results of a pre-study, which are presented in the previous section. Consequently, the duration parameter was fixed at 3.5 seconds for all mixer components. Furthermore, the Dynamic Parallel Mixer used a volume setting, which played-back the highlighted songs 200% louder than the background noise regarding a master volume of 40. The cross-fade parameter was set to 0.2 seconds. Furthermore, the music signals were normalized using the Linear Amplitude Normalization (see section 3.2.2) which received the required data from the Statistical Analyzer (root mean square and standard deviation) as described in section 3.2.1.

Each subject was confronted with 23 listening situations covering 11 different regions of the PlaySOM application which comprised areas assigned to the genres:

- Rock and Metal
- Electronic

ID	Mixer	Column Index	Row Index	#Songs	Genre
1	SPM	18–19	13	19	Classic
2	SPM	2–4	0–1	7	Electronic
3	SPM	4	2	5	Electronic
4	SPM	14	0	2	Classic
5	SPM	4–5	12	22	Rock and Metal
6	SPM	2–3	12	3	Rock and Metal
7	SPM	4	12	3	Rock and Metal
8	SPM	18–19	3–4	18	Classic
9	SPM	18–19	12	8	Classic
10	DPM	5	12	19	Rock and Metal
11	DPM	4	2	5	Electronic
12	DPM	14	0	2	Classic
13	DPM	2–3	12	3	Rock and Metal
14	SSM	0	1	15	Electronic
15	SSM	4	2	5	Electronic
16	SSM	2–3	12	3	Rock and Metal
17	SSM	18–19	12	8	Classic
18	SSM	18–19	3–4	18	Classic
19	DSM	18–19	3–4	18	Classic
20	DSM	18–19	12	8	Classic
21	DSM	4	2	5	Electronic
22	DSM	0	1	15	Electronic
23	DSM	2–3	12	3	Rock and Metal

Table 4.5: Detailed overview of all listening situations.

- Classic

Because of the high acoustic similarity of the Rock and Metal tracks covered by the test-set both genres were merged to the super-genre 'Rock and Metal'. Figure 4.5 shows the genre based segmentation of all listening situations. The 23 listening situations presented 9 areas assigned to Classic, 7 areas assigned to Electronic and 7 areas assigned to Rock and Metal. The 9 Classical regions covered 4 voice based and 5 instrumental oriented Classical areas. Table 4.5 illustrates all listening situations. It lists the position of the corresponding region in the PlaySOM application along with the mixer type used for the playback. Obviously various regions were presented several times using different mixer types. Prior to every listening situation each subject was shown where the presented region was located in the

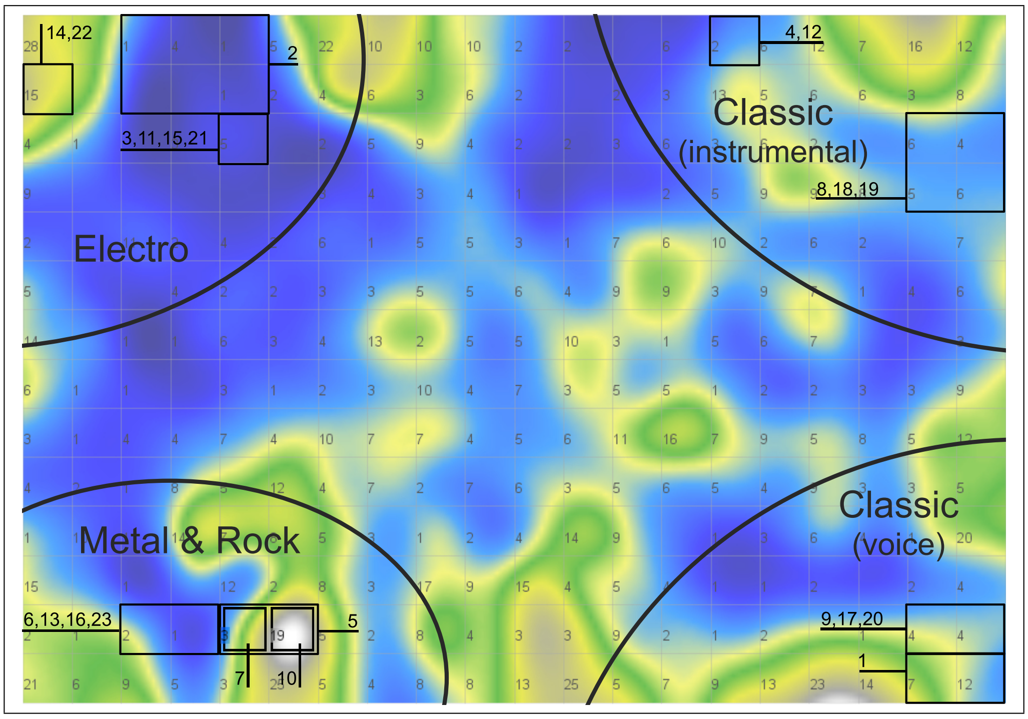


Figure 4.6: Every region of all listening situations marked in the music map.

music map. Figure 4.6 depicts each location of every listening situation in the music map of the PlaySOM application. After every listening situation each subject was interviewed using the questions listed in table B.2 in appendix B. Furthermore, every subject had to answer 11 over-all questions after the successful performance of all listening situations. This questionnaire is shown in table B.3 in appendix B.

All subjects were evaluated using the same main ordering of listening situations as listed in table 4.5. The participants were presented the listening situations based on the Static Parallel Mixer (SPM) first, followed by the ones using the Dynamic Parallel Mixer (DPM), the Static Sequential Mixer (SSM) and the Dynamic Sequential Mixer (DSM). The sequence of regions within this main organization changed between the subjects.

Results and Discussion

A very indicative characteristic of pieces of music is represented by singing parts. Because composers normally are interested in setting the human voice into the centre of compositions surrounded by instrumentation a high degree of voice identification can be assumed over all genres. This assumption is confirmed by the evaluation results, offering that nearly all subjects recognized existing singing components irrespective of the type of used mixer component. Figure 4.7 illustrates the averaged voice detection rate of all listening situations of a mixer component. Only the cognition of voice for the region covered by the listening situations 8, 18, 19 highly diverged. This area primarily consists of

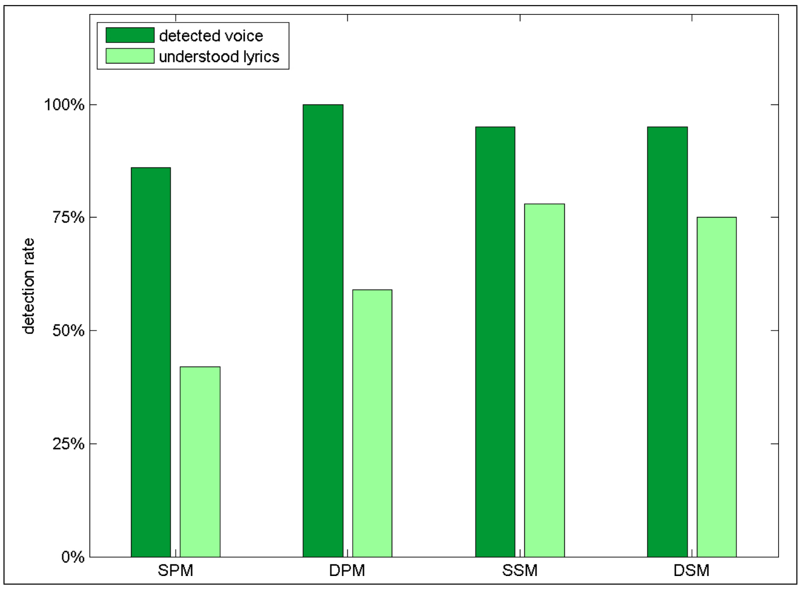
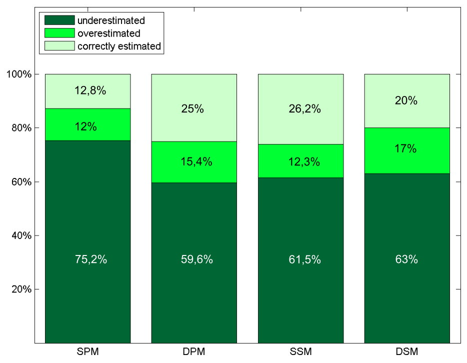


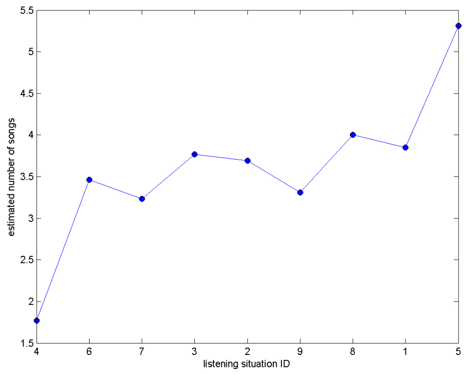
Figure 4.7: Voice detection and lyrics understanding according to all mixer types.

18 classical instrument-oriented songs but additionally contains a Rock piece with singing passages. Although 77% of the subjects perceived the singing parts when using both Sequential Mixer components only 8% were able to recognize a voice in the mix-down of the Static Parallel Mixer. In contrast to the high degree of correct voice identification over all listening situations, only few subjects were able to perceive the lyrics when using a Parallel Mixer instance. As depicted in figure 4.7 the degree of understanding is much higher when using one of the Sequential Mixer components.

A very difficult task for all subjects was estimating the perceived number of tracks during one listening situation. Especially listening situations based on the Static Parallel Mixer produced a mix-down which did not enable evaluators to reliably estimate the number of recognized songs. Nearly every participant shrugged their shoulders when being asked to estimate the perceived number of songs based on the mix-down of a Parallel Mixer instance. Interestingly subjects only rarely thought of having listened to more songs than there actually were. Only three subjects thought of having listened to more than 10 songs during a parallel playback scenario. The subjects estimated 72% of all possible scenarios smaller than 6 although only 47% definitely were. This result especially diverged for listening situations using the Static Parallel Mixer, where only 44% of the regions contained fewer than 6 songs. Anyway, the subjects estimated 84% of the areas containing fewer songs than 6. Table 4.5 delivers an insight of the number of songs covered by each listening situation. Although the subjects were not able to filter out the correct number of tracks during a parallel playback scenario the number they detected correlates with the number it actually was. Figure 4.8(b) illustrates the trend of the estimated number of tracks ordered according to the size of the presented region. Interestingly, although 87.2% were



(a) Distribution of estimated number of perceived songs for every mixer component.

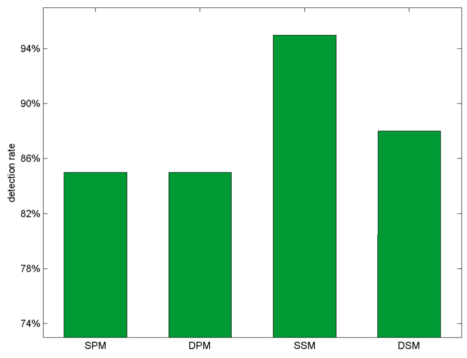


(b) Trend of averaged number of estimated songs according to the ordering in increasing region size.

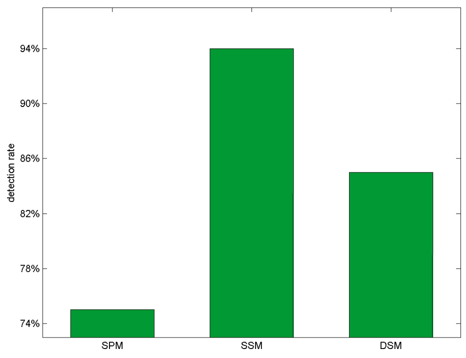
Figure 4.8: Trends of the estimation of the number of perceived songs.

incorrectly estimated the trend of the estimated scores corresponds to the trend of the underlying region's size. Generally, the participants undervalued the number of perceived tracks with decreasing divergence from the Parallel to the Sequence Mixer components. As depicted in figure 4.8(a) the best estimation results emerged when using the Static Sequential Mixer where 26% of the regions' size was correctly estimated. Remarkably, the use of the Dynamic Parallel Mixer conveys the listener an overview of the underlying data which nearly enables an estimation quality like the SSM does. Its performance of 25% correctly estimated numbers of songs is superior to the DSM.

A key aim of the evaluation was to analyze the subjects' ability in detecting the main genre of the presented region. While the detection of voice and estimation of region size was not influenced by the subjects' music knowledge and conventions of the real world, the assignment of a music mix-down to a main genre correlated with the participants' musical education and preferred genre. As depicted in figure 4.9(a) the subjects felt harder to correctly filter the main genre out of parallel playback than sequential playback. Figure 4.9(b) confirms the over-all trend of correct genre detection from parallel to sequential music playback based on four regions, all only used in combination with SPM, DSM and SSM. Interestingly, subjects who enjoyed a classical instrumental education, e.g. piano, were able to specify classical genres in more detail. They distinguished between e.g. Folklore, Choral, Church Music, etc. while others simply assigned the same areas to Classic. When confronting each of these subjects to this fact after having finished the evaluation process, they thought of simply assigning the area to Classic as too coarse and therefore false since all Classical music is not the same. A similar effect occurred when subjects were confronted with regions which contained pieces of their preferred genre. E.g., each subject was presented four listening situations covering the same area which



(a) Genre detection according to all 23 listening situations.

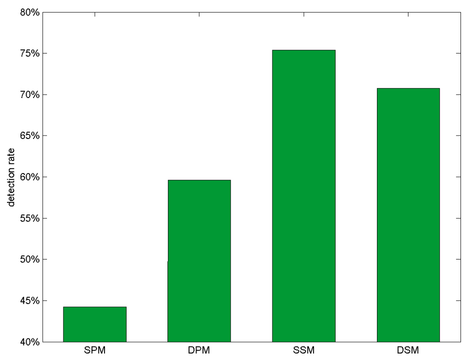


(b) Trend according to the listening scenario ID's 3, 6, 8, 9, 15, 16, 17, 18, 19, 20, 21, 23.

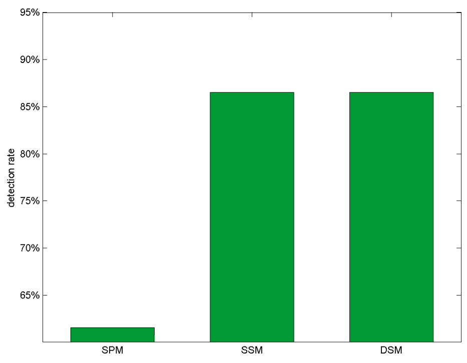
Figure 4.9: Ability of genre detection related to the mixer type.

contained three tracks of the genre Rock and Metal (ID's $6, 13, 16$ and 23). Only Subject6, who prefers the genre Punk, consistently over all different playback scenarios identified the main genre as Punk, while all others simply assigned the cluster region to Rock and Metal. Although Punk is a sub-genre of Metal, during the debriefing the subject mentioned that the tracks of this region never ever can simply be assigned to the genre Rock and Metal. In contrast to this accurate identification the same subject was not able to identify Classical regions in the same precise way.

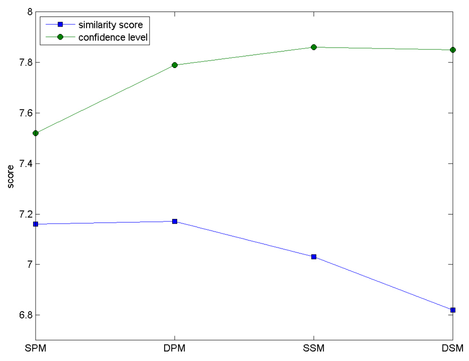
In parallel to this effect the outlier detection highly diverged between the subjects. Subjects detected non-fitting passages in areas consisting of tracks of their preferred genre more often. This indicates that subjects judge areas of their preferred genre stronger than other regions. E.g., the listening situations 14 and 22 are both based on a Sequential Mixer and cover the same region. The area contains 15 pieces of Electronic music. In order to compare the results of Subject9, obviously being a fan of Electronic music, with all 12 remaining subjects the results of Subject9 were extracted. Based on the mix-down of the Static Sequential Mixer only 23% of the remaining subjects thought that the region contains unfitting passages. Each of these participants detected exactly one single song which did not fit to the others. For the same region no one recognized a non-fitting piece during the listening situation based on the Dynamic Sequential Mixer. In contrast to that Subject9 detected four improper songs based on the SSM and three based on the DSM. Subject9 argued that the region consists of songs derived from completely different genres. As a consequence, Subject9's similarity assessment was much stricter compared to the remaining subjects. All remaining subjects rated the ground-similarity of the region



(a) Outlier detection rate according to all 23 listening situations.



(b) Trend according to the listening scenario ID's 3, 6, 8, 9, 15, 16, 17, 18, 19, 20, 21, 23.



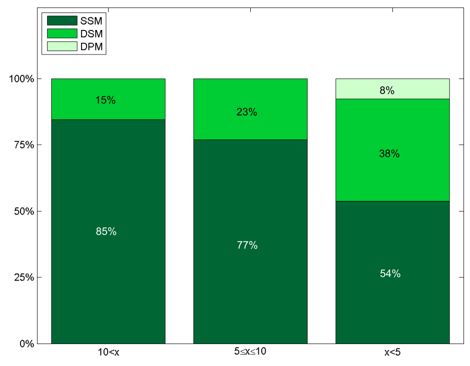
(c) Trend of similarity scores and confidence level.

Figure 4.10: Several trends from parallel to sequential playback.

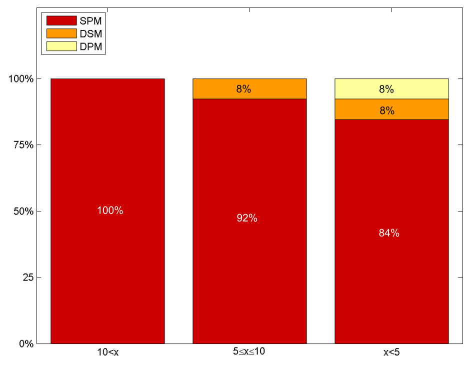
with 8.23[2] on average while rating their confidence level with 7.62[3] when using the Static Sequential Mixer. In contrast to that Subject9 rated the ground-similarity within the area with 4 with a confidence level of 10. This example indicates the trend that subjects perform a stricter judging for regions containing tracks of their preferred genre additionally to a more accurate differentiation between genre types. Consequently, the results yielded by judgments of areas of preferred genres might be of higher quality. However, most subjects argued for the detection of outliers with noticeably bothersome instruments in the music. The subjects' ability of detecting improper passages increased by switching from parallel playback to sequential playback. Figure 4.10(a) depicts this trend. While fewer than the half of all subjects detected an unfitting song during listening situations based on the Static Parallel Mixer, 75% of the participants thought of having perceived an improper song in a mix-down based on the Static Sequential Mixer. Although based on sequen-

[2] All participants were asked to rate the ground-similarity of the presented region (see question 9 in table B.2). A score of 1 denotes not similar and 10 means very similar.

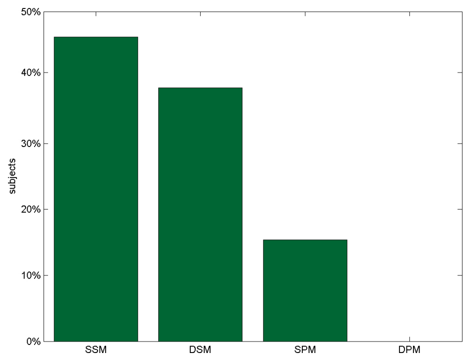
[3] A score of 1 denotes unsure and 10 means very sure.



(a) Mixer with best listening comfort based on a region containing x songs.



(b) Mixer with worst listening comfort based on a region containing x songs.

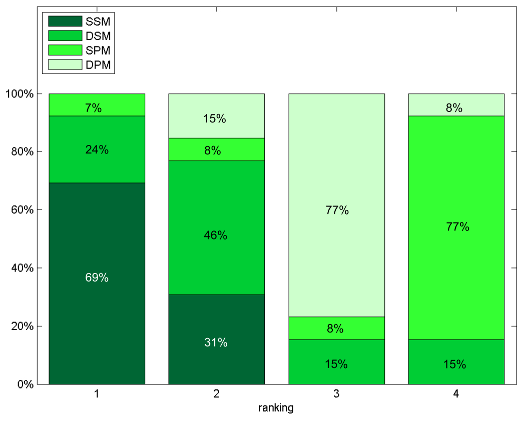


(c) Poll for the most effective mixer.

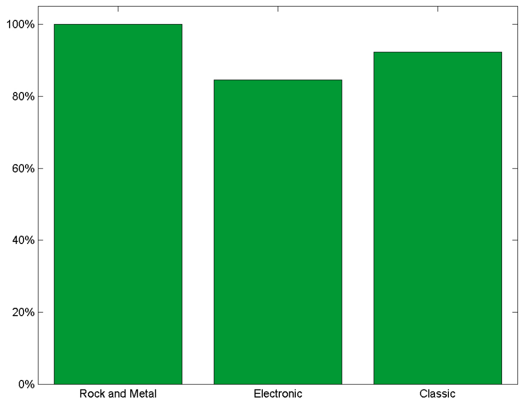
Figure 4.11: Subjects' perception of the different mixer types.

tial playback genre detection worked better, 60% of the subjects detected unfitting songs when using the DPM. The general trend is confirmed when analyzing four areas which all appeared in combination with the SPM, SDM and SSM which is illustrated in figure 4.10(b). Nevertheless, based on the SPM an outlier detection rate of 62% was achieved. Sequential playback enabled the subjects recognizing the presented region in more detail which resulted in stricter judgments. So the subjects were able to perceive more passages in the presented mix-down which do not fit to the rest of the signal. As a consequence, the subjects' rating of the ground-similarity decreased for sequential playback scenarios what is shown in figure 4.10(c). Nearly in parallel to the decrease of the similarity score the confidence level increased.

Most of the subjects seemed to be highly concentrated while listening to a mix-down performed by one of the Parallel Mixers. Many of them sat in a strained position, staring into space and frowning. During the interview session the subjects felt uncertain and needed quite a long time for answering a question. In contrast, being confronted with a sequential playback scenario the subjects seemed to be concentrated as well but without being overburdened with the mass of perceived information. Therefore, the participants



(a) Ranking of the different mixers according to the longest evaluation time without concentration loss.



(b) Reflection rate of presented genres.

Figure 4.12: Results of the over-all questionnaire presented in table B.3.

felt more confident during the interview situation and were able to respond within shorter time. This observation is confirmed by figure 4.11 which illustrates the subjects' perception according to best and worst listening comfort for a region consisting of a specified number of x tracks, i.e. 3 ranges with 1 to 4, 5 to 10 and more than 10. As depicted in figure 4.11(a) most of the subjects consider the Static Sequential Mixer to present a mix-down with the highest listening comfort. Interestingly, the consideration that the mix-down based on the Dynamic Sequential Mixer has the highest listening comfort raises with decreasing number of tracks although it should be independent from x. Figure 4.11(b) reflects the subjects' perception of which mixer type is linked with the worst listening comfort. Confirming the observation above the Parallel Mixer components are considered to produce a mix-down consisting of the worst listening comfort. Although nearly all subjects connected the Static Parallel Mixer with the worst listening comfort, about 15% of the subjects thought that it is the most effective one. However, figure 4.11(c) shows that most subjects assigned both of the Sequential Mixers to be most effective. Nearly all participants argued that indeed the playback time takes longer, but the perceived impression of the underlying data was significantly of higher quality. In correlation to these results 69% of the subjects think that the mix-down based on the Static Sequential Mixer enables them to perform an evaluation for the longest time without losing concentration, as illustrated in figure 4.12(a). Generally, the Sequential Mixer instances allow performing an evaluation for a longer time without a loss of concentration than the Parallel Mixers.

After the evaluation the subjects have been confronted with 23 listening situations covering 11 different regions of the PlaySOM music map. As already mentioned, prior to every listening situation each subject was made aware of the presented area's location in the music map. In order to analyze the intuitive structure of the visual representation of the test set the subjects were asked if they were able to spot coarsely the areas containing the genres Rock and Metal, Electronic and Classic. As depicted in figure 4.12(b)the corresponding areas of the three genres were correctly retrieved by the subjects with a performance of about 92%.

While the longest evaluation nearly took 55 minutes, all evaluations on average were finished after 45 minutes.

4.4 Summary

In this chapter two evaluations concerning tuning cLynx's Analysis Component and analyzing cLynx's applicability were presented.

The first study analyzed two different approaches for potentially raising the effectiveness of cLynx's Analysis Component by only partially analyzing the audio signal. The first method randomly extracted sample values which represented the basis for the calculation of a description value set Θ. The comparison to the original set Θ_{orig} figured out that the values resulting form the random approach were not sufficiently. The second method used uniform sample extraction by fixed interval steps and worked significantly better, with a deviation of only 6.2% when retaining 2% of the samples.

The second evaluation studied the human's ability of filtering music and judging music similarity based on the PlaySOM application and the cLynx plug-in. It showed that parallel playback has high accuracy in voice detection but does not enable subjects to detect the main-genre of listening examples. Additionally, the estimated number of perceived songs correlates with the size of the presented set of tracks which means that users get some sort of feeling of how many songs they currently evaluate. However, parallel playback demands a high degree of concentration which is unsustainable over a long period. The evaluation showed that sequential playback enabled the user a much better detection of genres. Tests pointed out that subjects performed a stricter judging of the similarity when the set of listening examples contained tracks of their preferred genre. This effect also occurred when subjects with musical education were confronted with listening examples consisting of Classical songs. Additionally, the degree of rigor in judging music similarity and assigning a song as outlier increased by switching from parallel playback to sequential playback.

Chapter 5

Conclusions and Future Work

In this thesis the problem of judging music similarity measures was described. Chapter 2 discussed basic signal processing methods and introduced into state-of-the-art techniques applied to systems which perform music organization based on similarity measures between audio tracks. An example system namely PlaySOM was presented which is extended by the plug-in cLynx, a tool offering novel approaches for music playback in order to support the user in subjective similarity judgment and outlier detection. Chapter 3 delivered a detailed insight into the design and functionality of cLynx. The user study, discussed in chapter 4, is based on a listening test in which subjects are confronted with acoustic mix-downs generated by the cLynx plug-in. The evaluation pointed out that subjective interpretation, individual preferences and previous musical knowledge correlate with the humans' perception of music similarity and susceptibility to outlier detection. Although all mixer types enabled the subjects to perceive singing parts, the results of the presented listening test justify that the subjects performed a significantly stricter outlier detection based on sequential playback than on parallel playback. By consequence, subjects were able to detect more unfitting passages during sequential playback scenarios. Nevertheless, based on the DPM an outlier detection rate of 60% was achieved. Additionally, both Sequential Mixers enabled better genre detection. However, based on the mix-down of one Parallel Mixer 85% of the presented areas were assigned to the correct genre. Even though the evaluation figured out that all mixers are useful for genre identification, sequential playback is recommended for scenarios containing many songs. Additionally, simultaneous playback is advised to get a coarse overview of the underlying audio data within short time which could be refined by using a Sequential Mixer afterwards.

The cLynx application which is documented in chapter 3 can be improved towards faster analysis, additional normalization and mixer component definition and improved user interaction. The following paragraphs point out several suggestions for possible improvements and useful extensions.

- Analysis time improvement: cLynx's most time-expensive task, the analysis component, could be improved by performing an analysis of only parts of the data, instead of processing a signal entirely. A thought-provoking impulse might be provided by the uniform analysis tuning approach which is described in section 4.2. Additionally, the random analysis tuning method, which only uses a defined number of parts of the signal, could improve its usefulness by implementing a heuristic. A possible and simple heuristic could force the algorithm to not only choose the parts randomly, but select some from the beginning, the middle and the end. This could eventually make the results more representative to the original values.

- Interactive Distance Mixer: The Distance Mixer presents the most immersive overview of a defined area in a cluster structure because of taking the spatial position of the signal within the area and its distance to the listener into account. As discussed in section 3.2.3 the Distance Mixer's problem is that signals which are placed very near to the origin (listener) can overlap other signals. Therefore, it would be advantageous to undock the listener from the origin and place her/him to different positions within the selected area. This would make signals accessible which are overlapped by a signal when the listener is placed in the origin. Additionally, the positioning of the signals could be implemented graphically in a similar way as described in [Latif and Mayer, 2007] extended by a graphical instance which represents the listener and allows to drag her/him to several positions on the plane.

Appendix A

Parameter Files

Table A.1 illustrates all keys along with their corresponding, possible values a valid start-parameter file must consist of. They are stored in the format of a simple Key-Value list like *key=value*. The parameter file can be passed using the parameter -sp in the execution command for the Graphical Stand-Alone Mode. For the Plug-In Mode cLynx's main class provides the constructor CLynx(Properties p) through which the path to the file can be passed.

Key	**Default**	**Possible Values**	**Description**
mixer	0	0 = Static Sequential Mixer 1 = Dynamic Sequential Mixer 2 = Static Parallel Mixer 3 = Dynamic Parallel Mixer 4 = Distance Mixer	Type of multiplexer
duration	3.0	Double value	Number of seconds a track is played back
fadeTime	0.3	Double value	Delay of a crossfade
channels	2	Integer value	Number of output channels for the Sequence Dynamic Mixer
volume	130	Double value	Volume for the Parallel Dynamic Mixer
volume.master	80	Double value	Master volume for the Parallel Dynamic Mixer
distancechannels	2	Integer value	Number of channels for the Distance Mixer
normalizer	2	0 = No Normalizer 1 = Linear Amplitude Normalizer 2 = Compressor	Type of Normalizer
compression	4:1	Integer:Integer	Ratio for the compressor
attack	50	Integer value	Attacktime for the Compressor
release	50	Integer value	Releasetime for the Compressor
analyser	3	0 = No Analyzer 1 = MinMax Analyzer 2 = Statistical Analyser1[a] 3 = Statistical Analyser2[b]	Type of Analyzer

[a]uses the arithmetic mean

[b]uses the root mean square

Table A.1: Possible keys and values of the start-parameter file.

Appendix B

User Study

ID	Question	Possible Answer
1	Did you recognize singing?	Yes, No
2	Would it be possible to partly understand the lyrics?	Yes, No
3	Did you recognize an Electronic beat?	Yes, No
4	Did you recognize special instrumentals?	Type of instruments
5	Did you recognize a dominant genre?	Name of genre
6	How many tracks did you recognize?	Number of songs
7	Did you hear songs not matching the main noise?	Yes, No
8	How many unfitting songs did you recognize?	Number of songs
9	How similar did you perceive all songs?	[1,10][a]
10	How did you feel the duration time?	too short (1), short (2), comfortable (3), long (4), too long (5)
11	Would a longer duration improve the quality of your impression?	Yes, No
12	How confident do you feel with your answers?	[1,10][b]
13	Did you sense the signals having constant equal volume level?	Yes, No

[a]1...unequal, 10...equal

[b]1...unconfident , 10...confident

Table B.1: Questionnaire of the pre-study.

ID	Question	Possible Answer
1	Did you recognize singing?	Yes, No
2	Would it be possible to partly understand the lyrics?	Yes, No
3	Did you recognize an Electronic beat?	Yes, No
4	Did you recognize special instrumentals?	Type of instruments
5	Did you recognize a dominant genre?	Name of genre
6	How many tracks did you recognize?	Number of songs
7	Did you hear songs not matching the main noise?	Yes, No
8	How many unfitting songs did you recognize?	Number of songs
9	How similar did you perceive all songs?	[1,10][a]
10	How confident do you feel with your answers?	[1,10][b]

[a]1...unequal, 10...equal
[b]1...unconfident , 10...confident

Table B.2: Questions each subject was asked after every listening situation.

ID	Question	Possible Answer
1	Which Mixer is the most comfortable for regions containing $>$ 10 songs?	Mixer
2	Which Mixer is the most uncomfortable for regions containing $>$ 10 songs?	Mixer
3	Which Mixer is the most comfortable for regions containing 5–10 songs?	Mixer
4	Which Mixer is the most uncomfortable for regions containing 5–10 songs?	Mixer
5	Which Mixer is the most comfortable for regions containing $<$ 5 songs?	Mixer
6	Which Mixer is the most uncomfortable for regions containing $<$ 5 songs?	Mixer
7	Which Mixer was the most efficient[a]one?	SSM (1), DSM (2), SPM (3), DPM (4)
8	Order the Mixer according to the longest concentration time when doing a long evaluation.	Mixer
9	Do you remember the region of the Music Map containing Rock and Metal songs?	North, South, West, East
10	Do you remember the region of the Music Map containing Electronic songs?	North, South, West, East
11	Do you remember the region of the Music Map containing Classic songs?	North, South, West, East

[a]The most efficient mixer represents the Mixer giving the subject an overview of the regions which enables the subject to answer the questionnaire presented in table B.2 with high confidence in less time.

Table B.3: Questions each subject was asked after finishing all listening situations.

Subject#:	1
Preferred Genre:	Electro
Age:	27
Sex:	male
Profession:	Real Estate
Instrument:	Piano

ID	**Q1**	**Q2**	**Q3**	**Q4**	**Q5**	**Q6**	**Q7**	**Q8**	**Q9**	**Q10**
1	1	0	0	-	Classic	3	0	0	6	2
2	0	0	1	Synthesizer	Electro	5	0	0	8	8
3	1	1	1	Synthesizer, Percussion	Electro	3	1	1	3	8
4	0	0	0	Guitar	Classic	1	0	0	10	9
5	1	0	0	Percussion	Rock & Metal	15	0	0	8	6
6	1	1	0	Piano	Rock & Metal	2	1	1	5	7
7	1	1	0	Percussion, Guitar	Rock & Metal	4	0	0	10	9
8	0	0	0	Piano, Guitar	Classic	3	0	0	9	9
9	1	1	0	-	Classic	3	1	1	8	8
10	1	0	0	Guitar, Percussion	Rock & Metal	13	0	0	9	9
11	1	1	1	Drums	Electro	4	1	1	6	8
12	0	0	0	Guitar	Classic	4	0	0	7	7
13	1	1	0	Piano, Percussion	Rock & Metal	4	1	1	7	8
14	1	1	1	Synthesizer	Electro	17	1	1	10	9
15	1	1	1	Synthesizer	Electro	5	1	1	9	9
16	1	0	1	Percussion, Guitar	Rock & Metal	4	1	1	8	9
17	1	1	0	-	Classic	14	1	1	9	9
18	0	0	0	Guitar	Classic	17	0	0	10	9
19	1	0	0	Guitar	Classic	8	1	1	9	9
20	1	1	0	Guitar	Classic	7	1	1	8	8
21	1	1	1	Drums, Synthesizer	Electro	6	1	1	7	8
22	1	1	1	Synthesizer	Electro	17	0	0	9	9
23	1	1	0	Guitar, Percussion	Rock & Metal	5	1	1	5	8

Q1	**Q2**	**Q3**	**Q4**	**Q5**	**Q6**	**Q7**	**Q8**	**Q9**	**Q10**	**Q11**
SSM	SPM	SSM	SPM	DSM	DPM	SSM	1-2-4-3	S	NE	SE

Table B.4: Results of Subject1

Subject#:	2
Preferred Genre:	Electro
Age:	31
Sex:	female
Profession:	Social Worker
Instrument:	-

ID	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10
1	1	1	0	-	Classic, Chorus, Church Music	2	1	1	6	8
2	0	0	1	Synthesizer	Electro	1	0	0	10	9
3	1	1	1	Clarinet	Electro	2	1	1	8	8
4	0	0	0	Stringed Instruments	Classic	1	0	0	10	10
5	1	0	0	-	-	-	-	-	-	2
6	1	1	0	Stringed Instrument, Percussion, Guitar, Keyboard	Pop	3	1	1	6	9
7	1	1	0	Guitar, Percussion	Rock & Metal	3	0	0	10	9
8	0	0	0	Harp	Classic	2	0	0	10	10
9	1	0	0	Flute	Classic	4	1	1	9	9
10	1	0	0	Guitar, Percussion	Rock & Metal	5	1	1	9	9
11	1	1	1	Percussion, Guitar, Keyboard	Pop	6	1	1	8	8
12	0	0	0	Harp	Classic	1	0	0	10	10
13	1	1	0	Stringed Instruments	Pop	4	1	1	7	8
14	1	1	1	Synthesizer	Electro	13	0	0	10	9
15	1	1	0	Synthesizer	Electro	3	1	1	7	8
16	1	1	0	Stringed Instrument, Percussion	Rock & Metal	3	1	1	2	8
17	1	1	0	Stringed Instruments, Flute	Classic	6	1	1	8	9
18	1	0	0	Stringed Instruments, Violin	Classic	4	1	1	9	9
19	1	1	0	Stringed Instruments, Flute	Classic	7	0	0	9	8
20	1	0	0	Stringed Instruments	Classic	4	1	1	9	9
21	1	1	0	Percussion, Guitar, Synthesizer	Electro	5	1	1	9	9
22	1	0	1	Synthesizer, Percussion	Electro	11	0	0	10	8
23	1	1	0	Stringed Instrument, Percussion, Keyboard	Rock & Metal	3	1	1	7	8

Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11
DSM	SPM	DSM	SPM	SSM	SPM	DSM	2-1-4-3	S	NW	E

Table B.5: Results of Subject2

Subject#:	3
Preferred Genre:	Jazz
Age:	31
Sex:	male
Profession:	Doctor of Medicine
Instrument:	Trumpet

ID	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10
1	1	0	0	Synthesizer	Classic	3	1	1	8	7
2	0	0	1	Synthesizer	Electro	3	0	0	3	5
3	1	0	1	Snare Drum, Bass Guitar, Synthesizer	Rock & Metal	6	1	3	2	2
4	0	0	0	Guitar	Classic	2	0	0	7	6
5	1	0	0	Piano, Guitar	Rock & Metal	3	1	1	8	8
6	1	1	0	Percussion, Guitar	Rock & Metal	4	1	1	5	9
7	1	0	0	Percussion, Guitar	Rock & Metal	3	0	0	8	9
8	1	0	0	Piano, Harp	Jazz	3	1	2	7	7
9	1	0	0	Strings	Jazz	1	0	0	10	5
10	1	0	0	Guitar, Contrabass	Rock & Metal	3	1	1	8	8
11	1	0	1	Percussion, Synthesizer	World	3	0	0	8	6
12	0	0	0	Guitar	Classic	1	0	0	10	10
13	1	0	0	Guitar, Harp	Pop	3	1	2	1	10
14	1	1	0	Synthesizer	Electro	4	1	1	9	8
15	1	0	1	Percussion, Synthesizer	Electro	4	1	1	5	5
16	1	1	0	Percussion, Guitar	Pop	3	1	1	6	6
17	1	0	0	Flute	Classic	4	1	1	8	6
18	1	0	0	Guitar, Piano	Classic	3	1	1	6	7
19	1	0	0	Percussion, Guitar, Piano	Classic	3	1	1	4	8
20	1	1	0	Flute	Classic	4	1	1	4	6
21	1	0	0	Bass Guitar, Percussion	Pop	3	1	1	7	5
22	1	1	1	Synthesizer	Electro	1	0	0	10	9
23	1	0	0	Guitar	Pop	3	1	1	6	7

Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11
SSM	DSM	SSM	SPM	DSM	SPM	SSM	3-1-4-2	SW	NW	SO

Table B.6: Results of Subject3

Subject#:	4
Preferred Genre:	Pop
Age:	28
Sex:	female
Profession:	Insurance Industry
Instrument:	Piano

ID	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10
1	1	1	0	Guitar	Choral, Church Music	3	0	0	8	8
2	0	0	1	Snare Drum	Electro	5	1	1	3	7
3	1	1	0	Percussion	World	8	1	3	6	7
4	0	0	0	Guitar	Folklore	2	0	0	10	8
5	1	0	0	-	-	2	0	0	0	10
6	1	1	0	Piano, Guitar	Rock & Metal	2	1	1	1	9
7	1	1	0	Guitar, Percussion	Rock & Metal	3	1	1	7	8
8	0	0	0	Harp, Violin, Stringed Instrument	Classic	3	1	2	8	7
9	1	1	0	Piano, Violin	Classic	2	0	0	8	8
10	1	0	0	Guitar, Percussion	Rock & Metal	2	1	1	3	7
11	1	0	1	Drums	Rock & Metal	4	1	2	7	7
12	0	0	0	Guitar	Folklore	2	0	0	10	9
13	1	1	0	Percussion, Guitar, Piano	Rock & Metal	2	1	1	2	9
14	1	1	1	Bass, Synthesizer	Electro	4	0	0	10	9
15	1	0	1	-	Electro	4	1	1	8	8
16	1	1	0	Guitar	Rock & Metal	3	1	1	6	7
17	1	1	0	Harp, Flute	Classic	3	0	0	10	8
18	1	0	0	Guitar, Piano	Classic	2	1	1	7	9
19	0	0	0	Guitar, Contrabass	Classic	2	1	1	7	8
20	1	1	0	Flute, Violin, Guitar	Classic	2	0	0	8	9
21	1	1	1	Percussion, Bass	Electro	2	1	1	5	8
22	1	1	1	Synthesizer, Bass	Electro	1	0	0	10	10
23	1	1	0	Percussion, Flute, Guitar	Rock & Metal	2	1	1	4	9

Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11
DSM	SSM	DSM	SSM	DSM	SSM	DSM	2-1-4-3	SW	NW	E

Table B.7: Results of Subject4

Subject#:	5
Preferred Genre:	Rock
Age:	30
Sex:	female
Profession:	Dancer
Instrument:	Piano, Flute

ID	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10
1	1	1	0	Guitar	Opera, Classic Choral	7	1	2	9	9
2	0	0	1	Synthesizer, Snare Drum	Electro	3	0	0	10	9
3	1	1	1	-	Electro	3	1	2	5	5
4	0	0	0	Plucked Instrument	Folklore	2	0	0	9	6
5	1	0	0	Violin	Classic	5	0	0	6	6
6	1	1	0	Guitar, Percussion	Rock & Metal	3	1	1	8	8
7	1	0	0	Guitar, Percussion	Rock & Metal	3	0	0	10	9
8	0	0	0	Piano, Violin	Classic	10	0	0	10	9
9	1	1	0	Contrabass, Piano, Bells	Classic	4	0	0	9	8
10	1	1	0	Guitar, Percussion, Piano	Rock & Metal	9	1	3	5	8
11	1	1	1	Guitar, Percussion, Synthesizer	Electro	4	1	1	8	8
12	0	0	0	Plucked Instrument	Folklore	1	0	0	10	9
13	1	1	0	Guitar, Percussion	Rock & Metal	4	1	1	9	9
14	1	1	1	Synthesizer	Electro	16	0	0	10	9
15	1	1	1	Synthesizer, Guitar	Electro	4	1	1	9	9
16	1	1	0	Guitar, Percussion	Rock & Metal	3	0	0	9	8
17	1	1	0	-	Classic	7	0	0	9	9
18	0	0	0	Plucked Instrument	Classic	13	0	0	10	9
19	1	1	0	Plucked Instrument, Violin, Piano, Harp	Classic	14	1	1	9	9
20	1	1	0	Organ, Flute	Classic	8	0	0	9	9
21	1	1	1	Synthesizer, Drums	Electro	4	1	1	9	9
22	1	1	1	Synthesizer	Electro	14	0	0	10	9
23	1	1	0	Plucked Instrument, Percussion, Guitar	Rock & Metal	3	1	1	8	9

Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11
SSM	SPM	SSM	SPM	DSM	SPM	SSM	2-1-4-3	S	NW	E

Table B.8: Results of Subject5

Subject#:	6
Preferred Genre:	Punk
Age:	26
Sex:	male
Profession:	Computer Scientist
Instrument:	-

ID	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10
1	1	0	0	Bass	Classic	2	1	1	1	7
2	0	0	1	Synthesizer	Electro	1	0	0	10	7
3	1	0	1	Synthesizer	Electro	2	1	1	3	4
4	0	0	0	Stringed Instrument	Classic	1	0	0	10	9
5	0	0	0	Guitar, Percussion	Rock & Metal	1	0	0	10	8
6	1	1	0	Piano	Punk	2	1	1	2	8
7	1	0	0	Bass, Percussion	Rock & Metal	2	1	1	7	7
8	0	0	0	Stringed Instrument	Classic	1	0	0	10	4
9	1	0	0	-	Classic	2	1	1	5	6
10	1	0	0	Percussion	Rock & Metal	2	1	1	1	6
11	1	0	0	Percussion	Pop	1	0	0	10	4
12	0	0	0	Stringed Instrument	Classic	1	0	0	10	8
13	1	1	0	-	Punk	3	1	1	7	5
14	1	1	1	Synthesizer	Electro	12	0	0	10	8
15	1	1	0	Synthesizer, Drums	Electro	6	1	1	8	7
16	1	1	0	Guitar, Bass Guitar	Punk	3	1	1	4	5
17	1	1	0	-	World	7	1	1	7	7
18	1	0	0	Guitar, Piano	Classic	10	1	2	9	8
19	0	0	0	Guitar, Piano	Classic	4	1	1	7	6
20	1	1	0	Flute	Classic	4	0	0	9	9
21	1	1	0	Synthesizer, Drums, Percussion	Pop	3	0	0	8	6
22	1	1	1	Synthesizer	Electro	1	0	0	10	5
23	1	1	0	-	Punk	3	1	1	4	5

Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11
SSM	SPM	SSM	SPM	SSM	SPM	SSM	1-4-2-3	S	W	NE

Table B.9: Results of Subject6

Subject#:	7
Preferred Genre:	Rock
Age:	26
Sex:	female
Profession:	Social Worker
Instrument:	Piano

ID	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10
1	1	1	0	Flute, Guitar	Classic	4	1	1	6	7
2	0	0	1	Synthesizer	Electro	3	0	0	8	8
3	1	0	1	Synthesizer	Electro	4	1	1	7	7
4	0	0	0	Guitar	Classic	2	0	0	9	8
5	1	0	1	Percussion	Rock & Metal	3	0	0	6	7
6	1	1	0	Guitar, Percussion	Rock & Metal	3	1	1	7	8
7	1	1	0	Percussion	Rock & Metal	2	1	1	2	8
8	0	0	0	Piano, Harp, Plucked Instrument	Classic	2	1	1	3	8
9	1	1	0	Flute	Classic	5	1	1	7	8
10	1	1	1	Percussion, Guitar	Rock & Metal	6	1	2	7	8
11	1	1	1	Synthesizer	Electro	4	1	1	7	8
12	0	0	0	Guitar	Classic	2	0	0	9	7
13	1	1	0	Percussion, Guitar	Rock & Metal	3	0	0	7	8
14	1	1	1	Synthesizer	Electro	18	0	0	9	8
15	1	1	1	Synthesizer, Drums	Electro	4	1	1	7	8
16	1	1	0	Percussion, Guitar	Rock & Metal	3	1	1	7	6
17	1	1	0	Plucked Instrument	Classic	7	1	2	7	7
18	1	1	0	Guitar, Piano	Classic	12	1	2	9	9
19	1	1	0	Guitar Flute	Classic	13	0	0	8	8
20	1	1	0	Bass, Transverse Flute	Classic	6	1	2	7	8
21	1	1	1	Synthesizer, Drums	Electro	6	1	2	7	7
22	1	1	1	Synthesizer	Electro	25	0	0	9	9
23	1	1	0	Handcuffs	Rock & Metal	5	1	1	7	7

Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11
SSM	SPM	SSM	SPM	SSM	SPM	SPM	1-2-4-3	SW	NW	E

Table B.10: Results of Subject7

Subject#:	8
Preferred Genre:	Jazz
Age:	28
Sex:	male
Profession:	Computer Scientist
Instrument:	-

ID	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10
1	1	0	0	Piano	Classic	8	1	1	7	10
2	0	0	1	Synthesizer	Electro	6	0	0	7	9
3	1	0	1	Synthesizer	Electro	2	1	1	4	10
4	0	0	0	Guitar	Classic	2	0	0	9	10
5	1	0	0	Snare Drums	Rock & Metal	12	0	0	0	4
6	1	0	0	Percussion, Guitar	Rock & Metal	6	1	1	8	9
7	1	0	0	Percussion, Guitar	Rock & Metal	4	0	0	8	10
8	0	0	0	Stringed Instrument	Classic	8	0	0	9	8
9	1	0	0	-	Jazz	3	1	1	6	10
10	1	0	0	Percussion, Guitar	Rock & Metal	8	0	0	7	5
11	1	0	1	Synthesizer, Percussion	Electro	6	1	2	6	7
12	0	0	0	Guitar	Classic	1	0	0	10	6
13	1	1	0	Percussion, Guitar	Rock & Metal	4	1	1	5	8
14	1	1	1	Synthesizer	Electro	10	0	0	6	8
15	1	1	0	Percussion, Guitar, Synthesizer	Electro	4	1	2	3	9
16	1	1	0	Stringed Instrument, Guitar	Rock & Metal	3	1	1	3	8
17	1	1	0	Flute	Classic	7	1	2	4	8
18	1	0	0	Guitar, Piano	Classic	8	1	1	6	7
19	1	0	0	Guitar, Piano, Percussion	Classic	12	1	3	7	6
20	1	1	0	Flute, Guitar	Jazz	7	1	3	5	6
21	1	1	1	Synthesizer, Percussion	Rock & Metal	5	1	2	4	7
22	1	1	0	Synthesizer	Electro	12	0	0	7	9
23	1	1	0	Guitar, Percussion, Flute	Rock & Metal	4	1	2	3	7

Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11
SSM	SPM	SSM	DSM	SSM	DSM	SPM	1-3-4-2	SW	NW	NE

Table B.11: Results of Subject8

Subject#:	9
Preferred Genre:	Electro
Age:	25
Sex:	male
Profession:	Computer Scientist
Instrument:	-

ID	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10
1	1	0	0	Violin, Flute	World	1	0	0	10	5
2	0	0	0	Synthesizer	Electro	5	1	1	7	9
3	1	0	1	-	Electro	4	0	0	6	7
4	0	0	0	Plucked Instrument	World	1	0	0	10	10
5	1	0	0	Percussion	Rock & Metal	2	0	0	7	6
6	1	1	1	Guitar, Percussion	Pop	3	0	0	7	5
7	1	0	0	Guitar, Percussion	Rock & Metal	1	0	0	10	6
8	0	0	0	Plucked Instrument	World	2	0	0	8	7
9	1	1	0	Guitar	World	1	0	0	10	6
10	1	1	0	Percussion	Rock & Metal	8	1	2	7	8
11	1	1	1	Synthesizer, Guitar	Electro	4	1	2	5	6
12	0	0	0	Guitar	Pop	1	0	0	10	4
13	1	1	0	Percussion	Pop	3	1	2	4	9
14	1	1	1	Synthesizer	Electro	15	1	4	4	10
15	1	1	1	Percussion, Keyboard	Electro	5	1	4	2	8
16	1	1	0	Percussion	Rock & Metal	3	1	2	3	9
17	1	1	0	Transverse Flute	Classic	8	1	2	6	6
18	0	0	0	Guitar, Piano	Classic	14	1	3	7	8
19	1	0	0	Guitar, Oboe	World	7	1	1	8	8
20	1	1	0	Flute	Classic	8	1	3	6	7
21	1	0	1	Synthesizer	Electro	8	1	1	6	7
22	1	1	1	Synthesizer	Electro	26	1	3	5	9
23	1	1	0	Percussion	Rock & Metal	4	1	1	7	9

Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11
SSM	SPM	SSM	SPM	DPM	SPM	SSM	1-4-2-3	S/SW	NW	SE

Table B.12: Results of Subject9

Subject#:	10
Preferred Genre:	Rock
Age:	22
Sex:	male
Profession:	Sport-Student
Instrument:	Guitar

ID	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10
1	1	1	0	-	Classic	4	0	0	9	9
2	0	0	1	Synthesizer	Electro	5	0	0	9	10
3	1	1	0	Synthesizer	Electro	4	1	1	7	8
4	0	0	0	Guitar	Classic	3	0	0	10	9
5	1	0	0	-	Rock & Metal	6	0	0	0	5
6	1	1	0	Guitar, Percussion	Rock & Metal	6	1	1	7	8
7	1	1	0	Guitar, Percussion	Rock & Metal	6	0	0	9	9
8	0	0	0	Stringed Instrument	Classic	4	0	0	9	9
9	1	1	0	Contrabass	World	4	0	0	7	7
10	1	0	0	Guitar, Percussion	Rock & Metal	9	0	0	8	9
11	1	1	1	Synthesizer, Percussion	Electro	5	0	0	9	9
12	0	0	0	Guitar	Classic	2	0	0	10	10
13	1	1	0	Guitar, Percussion	Rock & Metal	3	1	1	8	8
14	1	1	1	Synthesizer, Percussion	Electro	13	1	1	8	7
15	1	1	1	Synthesizer, Drums, Bass	Electro	5	1	1	7	8
16	1	1	0	Guitar, Percussion	Rock & Metal	5	1	1	7	9
17	1	1	0	-	World	6	1	5	2	7
18	1	1	0	Guitar, Piano	Classic	12	1	2	9	10
19	1	0	0	Guitar, Piano	Classic	8	1	2	7	8
20	1	1	0	Contrabass, Guitar	World	7	1	5	3	6
21	1	1	1	Synthesizer, Percussion	Electro	5	1	1	8	9
22	1	1	1	Synthesizer	Electro	13	0	0	9	9
23	1	1	0	Guitar, Percussion	Rock & Metal	4	1	1	7	8

Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11
SSM	SPM	DSM	SPM	SSM	SPM	DSM	1-2-3-4	SW	NW	NE

Table B.13: Results of Subject10

Subject#:	11
Preferred Genre:	Rock
Age:	22
Sex:	male
Profession:	Student in political science
Instrument:	Guitar

ID	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10
1	1	0	0	Bass Guitar	Classic	6	0	0	8	7
2	0	0	1	Synthesizer	Electro	3	0	0	8	7
3	1	0	1	Synthesizer	Electro	5	0	0	6	5
4	0	0	0	Guitar	Classic	3	0	0	8	8
5	1	0	0	Synthesizer	Rock & Metal	4	0	0	0	4
6	1	1	0	Guitar, Percussion	Rock & Metal	4	1	1	7	7
7	1	1	0	Guitar, Percussion	Rock & Metal	5	0	0	7	8
8	0	0	0	Stringed Instrument	Classic	4	0	0	8	7
9	1	1	0	Percussion, Keyboard	Rock & Metal	7	0	0	7	6
10	1	0	0	Guitar, Percussion	Rock & Metal	5	1	2	7	8
11	1	1	1	Synthesizer	Electro	4	0	0	8	7
12	0	0	0	Guitar	Classic	1	0	0	10	9
13	1	1	0	Guitar, Percussion	Rock & Metal	3	1	2	2	7
14	1	1	1	Synthesizer	Electro	12	0	0	6	6
15	1	1	1	Synthesizer, Guitar	Electro	4	0	0	7	7
16	1	1	0	Guitar, Percussion	Rock & Metal	3	1	1	2	7
17	1	1	0	Guitar, Harp	Classic	6	1	1	7	4
18	1	0	0	Guitar, Piano	Classic	10	1	2	8	8
19	1	0	0	Guitar, Percussion, Piano	Classic	7	1	2	7	7
20	1	0	0	Flute	Classic	6	1	1	6	7
21	1	1	0	Synthesizer, Percussion	Electro	4	0	0	6	7
22	1	1	1	Synthesizer	Electro	10	0	0	7	7
23	1	1	0	Guitar, Percussion	Rock & Metal	3	1	1	5	7

Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11
SSM	SPM	SSM	SPM	SSM	SPM	SSM	1-2-4-3	SW	SE	NE

Table B.14: Results of Subject11

Subject#:	12
Preferred Genre:	Rock
Age:	20
Sex:	male
Profession:	Student in agricultural science
Instrument:	Piano

ID	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10
1	1	0	0	Organ	Choral Classic	2	1	1	10	9
2	0	0	1	-	Electro	2	1	1	8	10
3	1	1	1	Percussion	Rock & Metal	2	1	1	10	8
4	0	0	0	Guitar	Folklore	1	0	0	10	8
5	1	0	0	-	Rock & Metal	1	0	0	10	4
6	1	1	1	Piano	Rock & Metal	2	1	1	1	9
7	1	0	0	Guitar	Rock & Metal	1	0	0	10	8
8	0	0	0	Piano	World	3	1	2	2	7
9	1	1	1	Strings	Classic	2	0	0	10	7
10	1	0	1	Guitar	Rock & Metal	3	1	2	8	8
11	1	1	1	Percussion	Electro	2	1	1	1	9
12	0	0	0	Guitar	Folklore	1	0	0	10	9
13	1	1	0	Guitar	Rock & Metal	3	1	1	6	7
14	1	1	1	Percussion	Electro	10	0	0	9	9
15	1	1	1	Percussion	Electro	5	1	1	6	7
16	1	1	1	Guitar, Percussion	Rock & Metal	4	1	1	2	8
17	1	1	0	Strings	Classic	4	0	0	9	5
18	1	0	0	Plucked Instrument	Folklore	2	1	1	7	7
19	0	0	0	Guitar, Piano	Classic	2	1	1	2	8
20	1	1	0	Strings	Classic	3	1	1	4	6
21	1	1	0	Percussion	Electro	2	1	1	2	10
22	1	1	1	Synthesizer	Electro	5	0	0	10	8
23	1	1	0	Guitar, Percussion	Rock & Metal	2	1	1	1	9

Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11
SSM	SPM	SSM	SPM	SSM	SPM	DSM	1-2-4-3	SW	NW	NE

Table B.15: Results of Subject12

Subject#:	13
Preferred Genre:	Jazz
Age:	26
Sex:	male
Profession:	Physiotherapist
Instrument:	Piano

ID	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10
1	1	0	0	-	Classic	5	0	0	9	6
2	0	0	1	Synthesizer	Electro	6	0	0	10	9
3	1	0	1	Synthesizer	Electro	4	1	1	6	7
4	0	0	0	Guitar	Classic	2	0	0	9	10
5	1	0	0	Percussion	Rock & Metal	15	0	0	9	5
6	1	1	0	Percussion, Guitar	Rock & Metal	5	1	1	7	9
7	1	0	0	Percussion, Guitar	Rock & Metal	5	0	0	8	6
8	0	0	0	Guitar, Piano	Classic	7	0	0	9	9
9	1	1	0	Organ, Piano	Classic	5	0	0	7	7
10	1	0	0	Percussion, Guitar	Rock & Metal	10	1	3	8	6
11	1	0	1	Synthesizer, Bass, Drums	Electro	5	1	2	6	8
12	0	0	0	Guitar	Classic	2	0	0	10	10
13	1	1	0	Percussion, Guitar	Rock & Metal	5	1	2	6	7
14	1	1	1	Synthesizer	Electro	5	0	0	10	9
15	1	1	1	Percussion, Guitar, Synthesizer	Electro	5	1	1	6	8
16	1	1	0	Percussion, Guitar, Flute	Rock & Metal	3	1	1	4	9
17	1	1	0	Organ	Classic	6	1	1	8	8
18	1	1	0	Guitar, Piano	Classic	6	1	2	8	9
19	1	0	0	Percussion, Guitar, Piano	Classic	5	1	2	7	9
20	1	1	0	Percussion, Guitar, Organ	Classic	8	1	2	6	8
21	1	1	1	Synthesizer, Guitar, Drums	Electro	5	1	2	5	6
22	1	1	1	Synthesizer	Electro	5	0	0	10	10
23	1	1	0	Percussion, Guitar, Flute	Rock & Metal	3	1	1	5	8

Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11
SSM	SPM	SSM	SPM	SSM	SPM	DSM	1-2-4-3	SW	NW	E

Table B.16: Results of Subject13

Bibliography

[A. Jain, 1997] A. Jain, D. Z. (1997). Feature selection: evaluation, application, and small sample performance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(2):153–158.

[Allamanche et al., 2003] Allamanche, E., Herre, J., Hellmuth, O., Kastner, T., and Ertel, C. (2003). A multiple feature model for musical similarity retrieval. In *Proceedings of the 4th International Conference on Music Information Retrieval (ISMIR 2003)*, pages 217–218.

[Aucouturier and Pachet, 2002] Aucouturier, J. and Pachet, F. (2002). Music similarity measures: What's the use? In *Proceedings of the 3rd International Conference on Music Information Retrieval (ISMIR 2002)*, pages 157–163.

[Bellman, 1961] Bellman, R. (1961). *Adaptive Control Processes: A Guided Tour.* Princeton University Press, Princeton.

[Berkhin, 2002] Berkhin, P. (2002). Survey of clustering data mining techniques. Technical report.

[Bishop, 1995] Bishop, C. (1995). *Neural Networks for Pattern Recognition.* Oxford University Press Inc., New York.

[Bock, 1974] Bock, H. (1974). *Automatische Klassifikation.* Vandenhoeck & Ruprecht, Göttingen.

[Brandes et al., 2003] Brandes, U., Gaertler, M., and Wagner, D. (2003). Experiments on graph clustering algorithms. In *Proceedings of the 11th European Symposium on Algorithms (ESA 2003)*, pages 568–579. Springer-Verlag, Berlin.

[Carreira-Perpinan, 1997] Carreira-Perpinan, M. (1997). A review of dimension reduction techniques. Technical Report CS-96-09, Dept. of Computer Science, University of Sheffield.

[Cox and Cox, 2000] Cox, T. and Cox, M. (2000). *Multidimensional Scaling.* CRC Press Inc., Boca Raton, 2nd edition.

[Cupchik et al., 1982] Cupchik, G., Rickert, M., and Mendelson, J. (1982). Similarity and preference judgment of musical stimuli. *Scandinavian Journal of Psychology*, 23:273–282.

[Dempster et al., 1977] Dempster, A., Laird, N., and Rubin, D. (1977). Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society, Series B*, 39:1–38.

[Dickreiter, 1997] Dickreiter, M. (1997). *Handbuch der Tonstudiotechnik*, volume 6. K.G. Saur Verlag KG, Munich.

[Dittenbach et al., 2005] Dittenbach, M., Neumayer, R., and Rauber, A. (2005). Playsom: An alternative approach to track selection and playlist generation in large music collections. In *Proceedings of the 1st International Workshop of the EU Network of Excellence DELOS on Audio-Visual Content and Information Visualization in Digital Libraries (AVIVDiLib 2005)*, pages 226–235.

[Dubes, 1987] Dubes, R. (1987). How many clusters are best? - an experiment. *Pattern Recognition*, 20(6):645–663.

[Fodor, 2002] Fodor, I. (2002). A survey of dimension reduction techniques. Technical Report UCRL-ID-148494, Lawrence Livermore National Laboratory.

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

[Gersho and Gray, 1992] Gersho, A. and Gray, R. (1992). *Vector Quantization and Signal Compression*. Kluwer Academic Publishers Group.

[Ghahramani, 2004] Ghahramani, Z. (2004). Unsupervised learning. In *Advanced Lectures on Machine Learning*, volume 3176 of *Lecture Notes in Computer Science*, pages 72–112. Springer-Verlag, Berlin.

[Gruzd et al., 2007] Gruzd, A., Downie, S., Jones, C., and Lee, H. (2007). Evalutron 6000: collecting music relevance judgments. In *Proceedings of the 2007 conference on Digital libraries (JCDL 2007)*, pages 507–507.

[Gurney, 1997] Gurney, K. (1997). *An Introduction to Neural Networks*. Taylor & Francis Ltd., London.

[H. Weisberg, 1970] H. Weisberg, J. R. (1970). Dimensions of candidate evaluation. *The American Political Science Review*, 64(4):1167–1185.

[Henle, 2001] Henle, H. (2001). *Das Tonstudio Handbuch*, volume 5. GC Carstensen Verlag, Munich.

[Horton, 2000] Horton, I. (2000). *Beginning Java 2*. Wrox, USA, jdk 1.3 edition.

[I. Borg, 2005] I. Borg, P. G. (2005). *Modern Multidimensional Scaling: Theory And Applications*. Springer Verlag, Berlin, 2nd edition.

[J. Gower, 1969] J. Gower, G. R. (1969). Minimum spanning trees and single linkage cluster analysis. *Applied Statistics*, 18(1):54–64.

[Jain and Dubes, 1988] Jain, A. and Dubes, R. (1988). *Algorithms for clustering data*. Prentice-Hall Inc., Upper Saddle River.

[Jain et al., 2000] Jain, A., Duin, R., and Mao, J. (2000). Statistical pattern recognition: A review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(1):4–37.

[Jain et al., 1996] Jain, A., Mao, J., and Mohiuddin, K. (1996). Artificial neural networks: A tutorial. *IEEE Computer*, 29:31–44.

[Jain et al., 1999] Jain, A., Murty, M., and Flynn, P. (1999). Data clustering: a review. *ACM Computing Surveys*, 31(3):264–323.

[Kaski et al., 1998] Kaski, S., Kangas, J., and Kohonen, T. (1998). Bibliography of self-organizing map (som) papers: 1981-1997. *Neural Computing Surveys*, 1:102–350.

[Kohonen, 1990] Kohonen, T. (1990). The self-organizing map. In *Proceedings of the IEEE*, volume 78 of *9*, pages 1464–1480. IEEE, New York.

[Kohonen, 2001] Kohonen, T. (2001). *Self Organizing Maps*. Springer Verlag, Berlin, Heidelberg, 3rd edition.

[Kruskal and Wish, 1978] Kruskal, J. and Wish, M. (1978). *Multidimensional Scaling*. Quantitative Applications in the Social Sciences. SAGE Publications Inc., Thousand Oaks.

[Latif and Mayer, 2007] Latif, K. and Mayer, R. (2007). Sky-metaphor visualisation for self-organising maps. In *Proceedings of the 7th International Conference on Knowledge Management (I-Know 2007)*, pages 400–407.

[Lidy, 2006] Lidy, T. (2006). Evaluation of new audio features and their utilization in novel music retrieval applications. Master's Thesis, Vienna University of Technology.

[Lidy and Rauber, 2005] Lidy, T. and Rauber, A. (2005). Evaluation of feature extractors and psycho-acoustic transformations for music genre classification. In *Proceedings of the 6th International Conference on Music Information Retrieval (ISMIR 2005)*, pages 34–41.

[Logan and Salomon, 2001] Logan, B. and Salomon, A. (2001). A music similarity function based on signal analysis. In *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME 2001)*, pages 745–748.

[Mitchel, 1997] Mitchel, T. (1997). *Machine Learning*. Mcgraw-Hill Professional, Columbus.

[Neumayer et al., 2005] Neumayer, R., Dittenbach, M., and Rauber, A. (2005). PlaySOM and PocketSOMPlayer: Alternative interfaces to large music collections. In *Proceedings of the 6th International Conference on Music Information Retrieval (ISMIR 2005)*, pages 618–623.

[Oja et al., 2003] Oja, M., Kaski, S., and Kohonen, T. (2003). Bibliography of self-organizing map (som) papers: 1998–2001 addendum. *Neural Computing Surveys*, 3:1–156.

[Pampalk, 2006] Pampalk, E. (2006). Computational models of music similarity and their application in music information retrieval. Dissertation, Vienna University of Technology.

[Pampalk et al., 2002] Pampalk, E., Rauber, A., and Merkl, D. (2002). Content-based organization and visualization of music archives. In *Proceedings of ACM Multimedia 2002*, pages 570–579.

[Pöllä et al., 2006] Pöllä, M., Honkela, T., and Kohonen, T. (2006). Bibliography of self-organizing map (som) papers: 2002–2005. unpublished manuscript.

[Raffaseder, 2002] Raffaseder, H. (2002). *Audiodesign*. Hanser Fachbuchverlag, Munich.

[Rauber et al., 2003] Rauber, A., Pampalk, E., and Merkl, D. (2003). The SOM-enhanced JukeBox: Organization and visualization of music collections based on perceptual models. *Journal of New Music Research*, 32(2):193–210.

[Ripley, 1996] Ripley, B. (1996). *Pattern Recognition and Neural Networks*. Cambridge University Press, Cambridge.

[Ritter and Schulten, 1988] Ritter, H. and Schulten, K. (1988). Kohonen's self-organizing maps: Exploring their computational capabilities. In *Proceedings of the IEEE International Joint Conference on Neural Networks (IJCNN 1998)*, pages 109–116. IEEE, New York.

[Scott, 1992] Scott, D. (1992). *Multivariate Density Estimation: Theory, Practice, and Visualization (Wiley Series in Probability and Statistics)*. Wiley-Interscience, Hoboken.

[Scott, 2002] Scott, D. (2002). *Principal Component Analysis.* Springer Verlag, Berlin, 2nd edition.

[Steinhausen and Langer, 1977] Steinhausen, D. and Langer, K. (1977). *Clusteranalyse - Einführung in Methoden und Verfahren der automatischen Klassifikation.* Walter de Gruyter, Berlin.

[Tzanetakis et al., 2001] Tzanetakis, G., Essl, G., and Cook, P. (2001). Automatic musical genre classification of audio signals. In *Proceedings of the 2nd International Symposium on Music Information Retrieval (ISMIR 2001)*, pages 205–210.

[Ultsch and Siemon, 1990] Ultsch, A. and Siemon, H. (1990). Kohonen's self-organizing feature maps for exploratory data analysis. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN 1990)*, pages 305–308. Kluwer Academic Publishers, Dordrecht.

[Vesanto, 1999] Vesanto, J. (1999). Som-based data visualization methods. *Intelligent Data Analysis*, 3(2):111–26.

[Vignoli and Pauws, 2005] Vignoli, F. and Pauws, S. (2005). A music retrieval system based on user driven similarity and its evaluation. In *Proceedings of the 6th International Conference on Music Information Retrieval (ISMIR 2005)*, pages 272–279.

[Warstat and Goerne, 1994] Warstat, M. and Goerne, T. (1994). *Studiotechnik: Hintergrund und Praxiswissen*, volume 6. Elektor Verlag GmbH, Aachen.

[Zahn, 1971] Zahn, C. (1971). Graph-theoretical methods for detecting and describing gestalt clusters. *IEEE Transactions on Computers*, C-20(1):68–86.

[Zurada, 1992] Zurada, J. (1992). *Introduction to Artificial Neural Systems.* Pws Pub Co.