FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

# Collaborative Music Consumption through Mobile Devices

DIPLOMARBEIT

zur Erlangung des akademischen Grades

**Diplom-Ingenieur**

im Rahmen des Studiums

**Software Engineering & Internet Computing**

eingereicht von

**Jakob Frank**
Matrikelnummer 0225733

an der
Fakultät für Informatik der Technischen Universität Wien

Betreuung
Betreuer: Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Andreas Rauber
Mitwirkung: Mag.rer.soc.oec. Rudolf Mayer

Wien, 22.03.2010

(Unterschrift Verfasser) (Unterschrift Betreuer)

Jakob Frank
Obere Amtshausgasse 14/29-30
A 1050 Wien

"Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe."

Wien, 22.03.2010

# Acknowledgments

I wish to thank Andreas Rauber for enabling me to work the field of Music Information Retrieval, for the challenges he presented me and his ongoing support and motivation.

I thank Rudolf Mayer and Thomas Lidy for the extensive discussions about scientific methods and technical solutions, and the late-night table tennis matches after a paper deadline.

I thank all colleagues and friends who accompanied me through my studies at the Vienna University of Technology.

# Zusammenfassung

Das immense Wachstum von verfügbaren Musiksammlungen in digitaler Form erfordern neue Zugriffs- und Interaktionsmöglichkeiten. Es existieren viele verschiedene Ansätze wie Musiksammlungen organisiert und durchsucht werden können, die aber nur selten über einfache Titellisten hinausgehen. Eine neue Herangehensweise, Playlisten einfach und schnell zu erstellen, bieten Musiklandkarten mit ihrem interaktiven und intuitiven Zugang zu großen Musiksammlungen. Sie erzeugen einen musikalischen Raum in dem durch die Auswahl einer Region oder durch Einzeichnen eines Pfades schnell und einfach Playlisten erstellt werden können. Dieser Ansatz soll in der hier vorliegenden Arbeit von unterschiedlichen Gesichtspunkten betrachtet werden.

Die Metapher, eine Playlist als Pfad durch einen musikalischen Raum zu sehen, soll im Zuge einer kleinen Benutzerstudie untersucht werden. Die Qualität von Playlisten aus unterschiedlichen Quellen wird dazu mit ihrer visuellen Darstellung im musikalischen Raum in Zusammenhang gesetzt, um so die Stärken und Schwächen der Metapher zu erkennen. Des weiteren sollen die Erkenntnisse dazu beitragen, die Qualität von automatisch generierten Playlisten zu verbessern.

Basierend auf der oben genannten Metapher werden verschiedene neue Ansätze zum kooperativen Musikgenuss präsentiert. Dabei müssen Vorschläge und Wünsche von unterschiedlichen Benutzern beachtet werden und in Einklang gebracht werden. Von den vielen verschiedenen Möglichkeiten werden einige mit ihren Vor- und Nachteilen gezeigt und diskutiert.

Um diese bisher theoretischen Konzepte auf eine praktische Basis zu setzten wird außerdem eine prototypische Anwendung vorgestellt, welche die Verwendung eines musikalischen Raums zur Erstellung von Playlisten auch auf mobilen Endgeräten ermöglicht. Des weiteren erlaubt die Anwendung die kooperative Verwendung des musikalischen Raums durch mehrere Benutzer.

Diese neuen Konzepte ermöglichen vielfältige neue Szenarien wie Musik in Zukunft konsumiert werden kann. Nicht nur der kooperative Ansatz erlaubt neuartige Anwendungen, auch eine allgegenwärtige Verbreitung vom musikalischen Räumen sind ein viel versprechendes Thema. Die Verschränkung von virtuellen und realen (musikalischen) Räumen kann speziell durch die Einbindung von mobilen Endgeräten erleichtert und verbessert werden.

# Abstract

The immersive growth of digitally available audio files calls for novel forms of interaction and consumption. There are many different approaches how to navigate and organize audio collections, but most of them do not offer much more than some kind of scrolling through lists, sometimes fancier, often not. A novel form of playlist creation is possible through *Music Maps*, which provide a interactive and intuitive interface to large music collections by creating a musical space where playlists are created by selecting regions on a map or by drawing trajectories on a map. This thesis discusses this theme from different points of view:

The *Path-Metaphor* of playlists on Music Spaces will be investigated to identify strengths and weaknesses of the underlying techniques used to create the above mentioned Music Maps. To achieve this, user generated playlists will be compared to playlists created with the metaphor and outstanding differences and common properties are calculated. This analysis is then used to identify the weaknesses of the Path-Metaphor. The information gained in this analysis process will further help to understand how, and based on which principles, human users create playlists. This will again help to improve the quality of automated created playlist.

Based on the Path-Metaphor, different approaches to enable *Collaborative Music Consumption* will be presented. Suggestions and requests from different users have to be considered when creating a collaborative playlist. There are many ways how to do this – some of them will be presented with their pros and cons.

To bring these, so far theoretical, concepts onto a practical basis, a prototype application for *portable devices* is then presented. This application will allow to use Music Maps on mobile devices by individual users and, to add the cooperative and multi-user aspect, provide different approaches of Collaborative Music Consumption.

The results of the so far mentioned work enables manifold scenarios of new ways of music consumption. Not only the collaborative and multi-user aspect opens new applications, but also the ubiquitous usage of musical spaces is a promising area. The combination of real and virtual musical spaces can be further propagated by including mobile clients which allow to quickly switch from the one setting to the other.

# Contents


**1 Introduction** **1**
- 1.1 Scenarios . . . 3
  - 1.1.1 Online Radio . . . 3
  - 1.1.2 Finding people with similar musical taste . . . 4
  - 1.1.3 Party Jukebox . . . 4
- 1.2 Contributions and Outline . . . 4

**2 Related Work** **6**
- 2.1 Audio Feature Extraction . . . 6
  - 2.1.1 Low-Level Audio Features . . . 7
  - 2.1.2 Mid-Level Audio Features . . . 7
  - 2.1.3 High-Level Features . . . 10
  - 2.1.4 MPEG-7 Standard . . . 10
  - 2.1.5 Feature Extraction Systems . . . 11
- 2.2 Music Organization . . . 12
  - 2.2.1 Embedded Metadata . . . 12
  - 2.2.2 Semantic Tags . . . 13
  - 2.2.3 Expert-based Cataloging . . . 14
  - 2.2.4 Collaborative Filtering . . . 15
  - 2.2.5 Map-Based Music Organization . . . 16
- 2.3 Social Music Consumption . . . 18
- 2.4 Automated Playlist Generation . . . 18
- 2.5 Applications . . . 21
  - 2.5.1 PlaySOM . . . 21
  - 2.5.2 MusicMiner . . . 23
  - 2.5.3 PocketSOM . . . 23
- 2.6 Summary . . . 27

**3 Creating and Manipulating Playlists** **28**
- 3.1 Autorouting . . . 28
  - 3.1.1 Routings . . . 29

3.2 Concatenation . . . 31
3.3 Intermediate Path . . . 32
3.4 Sub-Path Recombination . . . 34
3.5 Multiple Combinations . . . 34
3.6 Summary . . . 35

**4 ePocketSOM 37**
4.1 Implementation . . . 38
4.1.1 PocketSOM Connector . . . 41
4.1.2 Playlist Handlers . . . 44
4.2 Applications and Scenarios . . . 46
4.2.1 Map-based Music Player . . . 46
4.2.2 Remote Control . . . 46
4.2.3 Online Music Portal . . . 47
4.2.4 Autorouting . . . 48
4.2.5 Community Voted Playlist Generation . . . 48
4.3 Summary . . . 49

**5 Playlists as Paths or Trajectories in Music Spaces 50**
5.1 Data Corpus . . . 51
5.2 Visualizing Playlists . . . 53
5.3 User Study . . . 54
5.3.1 Participants . . . 56
5.3.2 Results . . . 57
5.4 Interpretation . . . 58
5.4.1 Radio (d) . . . 58
5.4.2 Radio (f) . . . 60
5.4.3 Last.fm (d) . . . 61
5.4.4 Last.fm (f) . . . 62
5.4.5 Routed . . . 64
5.4.6 Line . . . 64
5.5 Summary . . . 66

**6 Conclusions & Future Work 68**
6.1 Applications and Installations . . . 69
6.1.1 *Music Maps* in the Real World . . . 69
6.1.2 *Music Maps* in Virtual Environments . . . 69
6.1.3 Ambient *Music Maps* on Mobile Devices . . . 70
6.2 Outlook and Future Work . . . 70

**A User Study** **79**
A.1 Playlists . . . . . 79
A.2 Questionaire . . . . . 82
A.3 Answers . . . . . 85

# 1 Introduction

The pervasion of digital music calls for novel techniques for search, retrieval and access to music collections. Particularly mobile devices are, due to their limited display size and input capabilities, in a need of possibilities for intuitive and quick selection of music that go beyond mere browsing through song lists and directories.

In this thesis, a graphical user interface for mobile devices based on a *Music Map* is used for novel interaction approaches with audio collections. A *Music Map*, based on a Self-Organizing Map, organizes a music collection automatically by sound similarity through audio content analysis, provides an overview even over large audio collections, and offers several interaction possibilities to give users a quick and direct access to their music. It allows immediate creation of playlists based on music of a desired style or genre by pointing on clusters or drawing paths on the map. The application not only provides simple and intuitive access to music, but also enables novel application scenarios for community based music experience and collaborative music consumption. The system has been adapted for mobile devices, and implementations exist for a range of different platforms, such as PDAs and Smartphones. One implementation, based on the Eve-VM, a Java-based virtual machine, is part of this thesis (see Chapter 4).

The spreading of novel interaction forms together with the ubiquitous availability of music – on desktop PCs, home TV, mobile devices like PDAs and cell phones – will change the way music is consumed: In the age of digital music distribution, obtaining a specific piece of music is easier than ever.

Today's commercial music download platforms like iTunes[1] offer catalogs with more than 11 million songs[2]. The number of legal downloads of single audio tracks increased in Europe in the last years to over 1.5 billions in 2008 and is still increasing [IFP09]. Through the pervasion of 3G networks and Mobile Internet, the market for music that can be downloaded on mobile devices, e.g. the iPhone, has evolved and will soon become a major distribution channel not only for ring tones but also for full track downloads. Increasing bandwidth and lower data transfer costs encourage users to access digital content over their mobile devices, online music stores are

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

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

ubiquitously available.

Napster[3], an online radio stream and download service offers their songs any time and anywhere for consumers to enjoy music on the PC, their mobile phone or on an MP3 player. For a monthly flat rate customers can stream or download music online via stationary or mobile devices and reproduce their songs without limitations, as long as the monthly subscription is maintained.

Nokia provides a "*comes with music*"-website[4] where selected Nokia cell phones can access more than 5 million titles for download. Music from Nokia comes without DRM[5]-limitations and can be used on the mobile phone as well as on other devices. Remarkable is the business strategy of this service: The access fee for the *comes with music* service is paid together with and bound to the purchase of a supported Nokia cell phone. After buying the phone, the user has access to the music repository for one year, but can keep all downloaded music afterwards.

The impact of these developments to the customer's behavior is fundamental. Buying music increasingly switches from the experience of going into a record shop to flip through CDs and chat with the shop owner about new releases, towards online stores, which offer a virtually unlimited repertory of all different musical styles. Recommendations are often based on shop basket analysis or genre metadata. Moreover, music is more and more often sold on a per-track basis instead of precompiled collections and albums. No need to check whether the 15 other tracks on the album are worth buying, you just pick the song you like and that's all you get. Services like TrackID[6] or Shazam[7] can further help to identify a song that is currently played on the radio or in a bar: an application on the mobile phone records a short sample of the song, queries an online database and prompts track title, artist and album name together with a link to purchase the track in an online store.

But not only the access, also the interaction with music is changing. Online platforms like last.fm[8] allow people from different regions in the world to chat and exchange about music. Online radio streams together with social networks and instant messaging allow listening to music in a group beyond geographic constraints. It is only a small step further to private audio streams shared to a group of friends over the Internet based on the collective audio collections of all participants with an integrated instant messenger.

---

[3] `http://napster.com`

[4] `http://www.comeswithmusic.de/`

[5] Digital Rights Management

[6] `http://www.sonyericsson.com/cws/support/phones/detailed/whatistrackid`

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

[8] `http://last.fm/`

While accessing music gets technically easier, the sheer amount of information makes it more and more difficult to actually *find* music. Retrieving a song by artist name and title is rather simple, but currently online stores lack a satisfying way to roam through their repertory without specifically knowing an artist or song name one would like to find. This opens the field for a number of services that help customers to keep track of the music available or to filter the repertory to a manageable number.

Map-based interfaces can provide an intuitive overview to a large data collection based on audio similarity. Especially on mobile devices with limited interaction possibilities such novel interfaces can increase the usability and accessibility of large data collections, such as digital music libraries which are nowadays present on virtually every mobile device. Map-based interfaces to digital music collections further allow new interaction with the collection such as creating playlists based on a path metaphor, for single users or in a group. Moreover, *Music Maps* can provide a playful access to online music stores and streaming services.

Several approaches to combine playlists based on the path metaphor are shown in this thesis which allow to create a playlists based on contributions from different users, or to merge several audio streams into one thus saving server load. The quality of playlists from different sources, including playlists created based on the path metaphor, are evaluated in a small scale user study and a mobile client, providing an alternative interface to large music databases, with multi user interaction concepts is presented.

## 1.1 Scenarios

The techniques presented in this thesis enable new forms of interaction and business models in many different ways. In the following, some of them are briefly outlined to give an insight into the potentials.

### 1.1.1 Online Radio

One scenario where the techniques presented in this thesis might be useful is an online personalized radio station.

This radio station could offer three different ways to select a radio channel:

1. By navigating to a certain region on the map, users can attach themselves to existing radio channels and by this follow other users on their way over the map. By additional social network features, users who listen to the same channel can then comment on the current song, chat with each other and more.

2. Users can create or join a group to collaboratively create a new channel. The playlist of this channel is generated based on the different playlists submitted by the participants.

3. Individual channels are created by a single user who can decide whether he/she wants to allow other users to attach themselves to this channel or not.

### 1.1.2 Finding people with similar musical taste

People sharing the same musical taste tend to also share other interests. This means that people listening to music from the same region on a *Music Map*, might also share other interests.

The idea is to bring people together that repeatedly explore the same regions on the map, e.g. by providing a chat that delivers messages to all users within a defined range on the map, or other social network activities.

### 1.1.3 Party Jukebox

Creating the playlist for a party is difficult, since wishes and music taste of many different people have to integrated into one playlist. Different approaches to handle this situation are possible, the simplest is to just activate "shuffle" – let the player application decide. Other possibilities are to create a playlist in advance or to ask someone to take over the DJ-job for that evening.

A complete different possibility is to delegate the creation of the playlist to all guest of the party. The system is based on *Music Maps*, where guests can deposit their wishes by either selecting a region or drawing a trajectory on the map, or by picking individual song from the repository. The system then condenses these inputs to a combined playlist. A range of methods to achieve this is presented in Chapter 3.

## 1.2 Contributions and Outline

These main contributions of this thesis consist of

- Research into new techniques to create, merge and manipulate playlists based on the path metaphor on *Music Maps*. Specifically, a set of techniques is introduced to combine playlists from different users into one joint playlist.

- Development of novel concepts for multi user access to and interaction with *Music Maps*. These allow multiple users to share their paths and playlists on a

central *Music Map*, and collaboratively create a playlist by combining them to a joint playlist.

- The implementation provides the techniques listed above on mobile devices running the Eve VM, and has been intensively tested on a range of mobile devices. Is is freely available for download at the projects website[9].
- Significant effort has been invested in evaluating the concepts of playlists in different settings, with a specific focus on the concept of similarity between subsequent songs in playlists, and their correspondence to the organizing principles underlying the SOM-based *Music Maps*.
- The resulting system was evaluated in a user study evaluating the quality of playlists from different sources, including automatically generated playlists.

Several parts of the work done for this thesis have been reviewed and published at international scientific conferences: Different stages of development of the mobile client to remotely access *Music Maps* have been presented in [NFH+07, SWV+08, FLHR08], whereas [FLP+09] shows the impact of the client on ambient music consumption. Early fundamental concepts to merge and manipulate playlists based on the path metaphor were presented in [Fra08]. In [Fra09], preliminary results of the user study were published. The system has also been presented at international fairs such as CeBit and IBC, and at several international tutorials and other showcase events.

The remainder of this thesis is structured as follows: Chapter 2 gives an overview about related work in this field and explains the technical background of the system. Chapter 3 presents different methods of playlists manipulation based on the path metaphor on *Music Maps*.

In Chapter 4 the implementation of *Music Maps* based on the Eve-VM for mobile devices is presented together with its additional multi-user features. Chapter 5 investigates the correlation between the quality of playlists and their visual shape on a *Music Map* based on a small scale user study.

Finally, Chapter 6 draws together the implications and lists use cases, emerging applications, and topics of future investigation.

[9] `http://www.ifs.tuwien.ac.at/mir/pocketsom/ewesom/`

# 2 Related Work

This chapter gives an overview on related work from different audio research teams. Section 2.1 will give a short introduction into different feature sets such as MFCC and Rhythm Patterns. Section 2.2 presents various possibilities to organize audio collections, and Section 2.3 describes approaches for collaborative music consumption. Section 2.4 introduces algorithms and methods for automated playlist generation, while Section 2.5 references applications that provided the basis of *ePocketSOM* presented in Chapter 4.

## 2.1 Audio Feature Extraction

Audio Feature Extraction is the process of deriving descriptive information from audio files or streams. Many different algorithms have been developed in the domain of musical feature extraction. They can be roughly divided into two groups, depending on the input source used to analyze the music: A symbolic representation, e.g. used in MIDI files, or wave-form based signals, as e.g. WAV or MP3 files.

A range of feature extraction algorithms use a symbolic notation of music as input source. This structured and well defined input data can simplify the analysis. As such, analyzing or extracting a specific instrument, determining the rhythm or chord detection are rather easy since all important information are directly written in the input data.

The major disadvantage with symbolic music is the availability of data files. MIDI files generally are rare and hard to find in particular for modern mainstream music, while files containing the mixed audio-signal, especially in times of online music stores, are wide spread and available for almost every track. Furthermore, while the perception of MIDI files strongly depends on the synthesizer (as much as the perception of scores depends on the musician playing them), MP3- or WAV-files are replayed "as-is" apart from effects of an equalizer or the speakers.

There are several approaches to combine both, symbolic and audio feature algorithms by transforming the data from one representation to the other. While this technique works quite well for MIDI-to-audio, the other way round is complicate, especially for polyphonic music and is still a hot topic in current research projects.

Because the main focus of this work is based on mixed audio-signal files, also the focus of feature extraction algorithms will be on audio features.

### 2.1.1 Low-Level Audio Features

Low-level features are directly derived from the audio wave signal and are used in many content-based audio retrieval projects. Some exemplary features will be briefly described here. [TC99]

The *Zero Crossing Rate* is a rather simple feature, representing the number of times the signal crosses the 0-line within one second. It serves as a basic separator of speech and music.

The *Silence Descriptor* expresses the duration of silence in the audio signal. It can be used as basic separator for speech and music, or to find track changes in a continuous audio stream.

*Root Mean Square Energy* (RMS) and *Low Energy Rate* are two features based in the time domain. RMS gives a good indication of loudness in a time frame and may also serve for higher-level tasks such as beat estimation. It is calculated by the mean of the square of all sample values in a time frame and taking the square root.The Low Energy Rate describes the percentage of frames containing a lower RMS than the average RMS of all frames.

The *Spectral Flux*, *Spectral Centroid* and *Spectral Rolloff* are based in the frequency domain. Spectral Flux describes the spectral change i.e. between two frames. The Spectral Centroid is the frequency where the summed energy below is equal to the summed energy above that frequency and is a measure of general spectral shape and brightness. Spectral Rolloff is a measure of skewness of the spectral shape.

### 2.1.2 Mid-Level Audio Features

Mid-level features contain further aggregation and calculation than low-level features.

The *Beat Histogram* [TC00a] is a rhythm periodicity function representing beat strength and rhythmic content, of which the following feature sets can be derived: the relative amplitude, the amplitude ratio and the period (as beats per minute) of each the first and second peak, and the overall sum of the histogram as indication of the beat strength.

The *Pitch Histogram* [TC00a] is computed by decomposing the signal into two frequency bands. For both bands, the amplitude envelope is extracted and summed up, and the dominant pitches are detected. The three dominant peaks are summed up into the histogram, each bin corresponding to a musical note. This histogram contains information about the pitch range. A folded version of the histogram where the notes

of all octave are mapped into a single octaves contains information about the pitch classes and the overall harmonic. The following features are derived from the Pitch Histograms: from the unfolded histogram the period of the maximum peak (the dominant octave) and the overall sum as the strength measurement of pitch detection. From the folded histogram the period of the maximum peak (the dominant pitch class), the amplitude of the maximum peak (the magnitude of the dominant pitch class) and the pitch interval between the to most dominant peaks (the main tonal interval relation).

*Mel Frequency Cepstral Coefficients* (MFCCs [Log00]) are perceptually based spectral features originally developed for speech processing. A cepstrum is the inverse Fourier transform of the logarithm of the spectrum. The frequency bands are mapped to the Mel scale, an approximately logarithmic scale corresponding closely to the human auditory system before applying the inverse Fourier transform. In most cases, only the first twelve coefficients are used.

*Rhythm Patterns* (RP), *Rhythm Histogram* (RH) and *Statistical Spectrum Descriptors* (SSD) are audio features describing the rhythmic structure of the audio signal, also considering psycho-acoustic models that represent the specific characteristics of the human hearing [RF01, LR05, Lid06]. As these features will be used for the experiments presented in this theses, a more detailed discussion of their computation is provided.

The calculation of the audio features is done in several steps: the first step is to calculate the specific loudness sensation in different frequency bands (see Phase 1 in Figure 2.1). This is done by a STFT (Short-time Fourier Transform) where the resulting frequency bands are grouped to psycho-acoustically motivated critical-bands based on the Bark Scale [Zwi61]. Then spreading functions are applied to account for masking effects and followed by a transformation into decibel, Phon and Sone scales. The result is a power spectrum that reflects human loudness sensation (Sonogram).

By calculating various statistical moments (mean, median, variance, skewness, kurtosis, min- and max-value) from the Sonogram, the *Statistical Spectrum Descriptor* features are computed.

By applying a second Fourier Transform (Phase 2 in Figure 2.1) the power spectrum is transformed into a time invariant representation based on the modulation frequency. The result are amplitude modulations of the loudness in individual critical bands. From these Modulation Amplitude Spectrum the *Rhythm Histogram* is computed by aggregation. Depending on their frequency, amplitude modulations have different effects on human hearing sensation. From this representation, reoccurring patterns in the individual critical bands, resembling rhythm, are extracted, which – after applying Gaussian smoothing to diminish small variations – result in a time-invariant, comparable representation of the rhythmic pattern in the individual critical bands: the *Rhythm Pattern*.

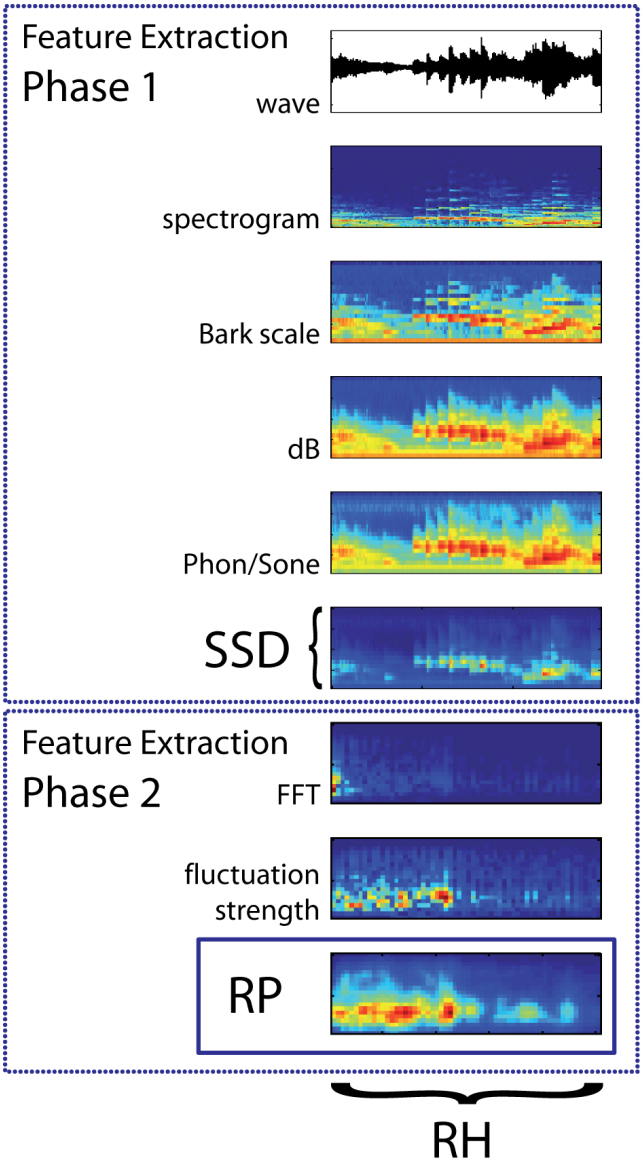



Figure 2.1: Overview of the extraction process for RP, RH and SSD

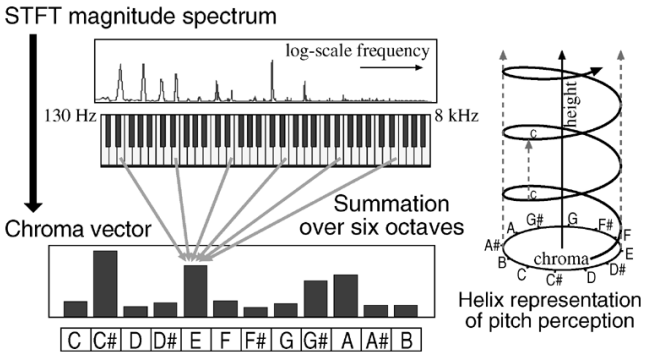



Figure 2.2: Overview of how a chroma vector is calculated. The Shepard's helix is depicted at the right. [Got06]

### 2.1.3 High-Level Features

High-level features express a certain musical concept such as a chord, or a melody.

*Chroma Features*, or a chroma vector, is a perceptually-motivated feature vector based on the concept of *chroma* in the Shepard's helix representation [She64]. Chroma refers to the position of a musical pitch within an octave that corresponds to a cycle of the helix (see Figure 2.2 at the right).

The chroma vector represents magnitude distribution that is folded into one octave and discretized and summed up into twelve pitch classes corresponding to the twelve notes in a standard chromatic scale (i.e. the twelve black and white keys within one octave on a piano). A schema of this process is depicted left-hand in Figure 2.2. Chroma vectors are able to capture the overall harmony (the pitch class distribution) and have proven to be effective identifying chord names. [YKK+04, Got06]

Many more high level features exist like descriptors for key, mode and tonality. A extensive overview is given in [Dow03] and [Ori06].

### 2.1.4 MPEG-7 Standard

The MPEG-7 standard [Mar04] is a standard for the description of audio and visual content, developed by MPEG (Motion Picture Experts Group) a working group if ISO/IEC developing standards for digitally coded representation of audio and video[1]. The group has produced several wide-spread standards such as MPEG-1, which is the basis for the famous MP3-format or MPEG-2 and MPEG-4 which are used for video compression.

[1] `http://www.chiariglione.org/mpeg/`

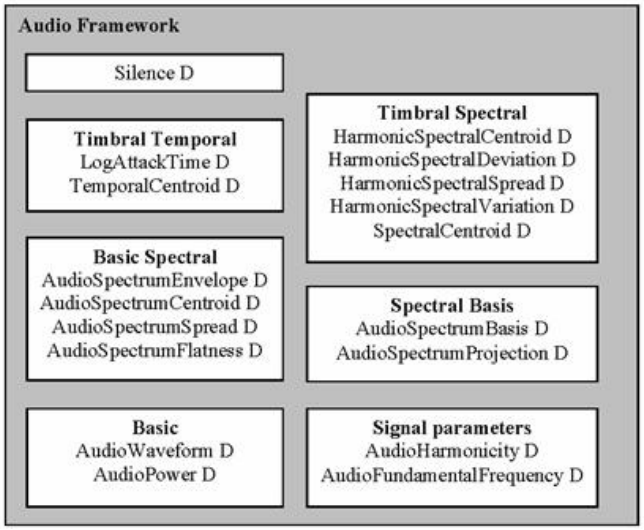



Figure 2.3: Overview of features in the MPEG-7 Audio Framework [Mar04]

The *MPEG-7 Audio Framework*, part of the MPEG-7 standard, comprises 17 low-level features and the *High-level audio Description Tools*, which are based on the low-level features and designed for specific applications (e.g. instrument timbre and spoken content). The 17 low-level features are grouped into the following seven classes: *Basic Temporal Descriptors* and -*Spectral Descriptors*, *Signal Parameters*, *Timbral Temporal Descriptors* and -*Spectral Descriptors*, *Spectral Basis Descriptors* and the *Silence Descriptor*. An overview of the groups and the features contained therein is depicted in Figure 2.3.

A brief review of these classes and the individual features can be found in [Lid06], the complete standard is available online[2]

### 2.1.5 Feature Extraction Systems

There exists a wide range of different software packages extracting different sets of features.

The *MARSYAS system* [TC00a, Tza02] is an open source software framework for audio analysis, feature extraction and retrieval. It provides a number of feature extractors that can be divided into three groups: features describing the timbral texture, those capturing the rhythmic content, and features related to pitch content. The rhythm-related features express regularity in rhythm and are based on the *Beat Histogram*, the

[2] `http://www.chiariglione.org/mpeg/standards/mpeg-7/mpeg-7.htm`

pitch related features are derived from *Pitch Histogram* (see Section 2.1.2).

Furthermore, Tzanetakis presented an approach to extract audio features directly from MPEG compressed audio files (e.g. from MP3 files) [TC00b]. The basic idea is that during the MPEG compression much of the analysis is already done. Instead of decoding the data and again compute the spectrum this approach calculates the features directly from the MPEG data.

The *Matlab Auditory Toolbox*[3] is a set of popular feature extraction algorithms implemented in Matlab. It provides a huge number of feature extraction algorithms, low-level as well as high level features, which can be easily combined to create and evaluate new feature sets.

*M2K* (Music-to-Knowledge) [DEH05] is a tool set written in Java for developing and evaluating algorithms and techniques in the field of music information retrieval. It provides a default set of often used feature extraction algorithms such as MFCC but also higher level features like onset detection.

*CLAM* (C++ Library for Audio and Music) [AAG06] is a library for music and audio analysis. It comes with a set of feature extraction algorithms such as low level spectral features and chord detection among many others. Additional modules for new algorithms can be added to the library easily.

## 2.2 Music Organization

Music collections can be organized in many various ways, each suitable for a different task. The following sections will briefly introduce some approaches.

### 2.2.1 Embedded Metadata

Digital music files frequently contain, in addition to the audio signal, related meta-data, which may contain textual descriptive information (e.g. song title, album, artist, genre, year) but also graphical data (e.g. an album cover), lyrics or other arbitrary additional data (e.g. audio features, see Section 2.1). These descriptive information consists of key-value pairs, e.g. artist="W. A. Mozart". A wide spread method of adding embedded meta-data to MP3-audio files are ID3 tags which first came up in the mid-1990s. The current version ID3v2 [Nil99] specifies many additional information tags that can be inserted into an audio file, however, most implementations only support simple text based tags for title, album, artist, and some more. Advanced tags

[3] `http://users.jyu.fi/~lartillo/mirtoolbox/`

like album images are only rarely supported and thus very infrequently used. Embedded metadata-tags must not be mixed up with semantic tags that are often used in web-based portals (see Section 2.2.2).

Tags constitute a major improvement over searching for file-names only. If a search interface enables searching for different tags users have much more freedom issuing their search queries. Using tag related search, finding tracks from a specific album or by some artist is very easy.

Since most of the tagging is done by humans, the information might be inconsistent. Different individual users might associate different tags to the same track. Especially ambiguous tags like genre are hard to determine and will differ from user to user, also depending on their personal experience [LPA03]. For instance, if a user is searching for music from the genre *Alternative Rock* in some music library, the result might contain songs he would never associate with this category. But also presumably well defined tags such as artist or track title frequently differ because of various, more or less creative spelling flavors users might invent, especially for less common names, or names containing special characters or just by typing mistakes.

### 2.2.2 Semantic Tags

A semantic tag is a keyword or term attached to an item, in our case attached to a song or an audio file. Contrary to metadata, semantic tags are not from a predefined collection of fields (e.g. artist, title, album) but may contain arbitrary terms describing the tagged item. Last.fm[4], for example, is an online music platform where semantic tags are heavily used. Users can attach free text tags to any song, artist or album giving a semantic and social context to the item. These tags can be used to search for a song or artist previously unknown. [Lam08]

The idea of semantic tags is not new, the technique has been used for years in various services. Delicious[5], founded in 2003, is a social bookmarking service that allows users to save their bookmarks in an online platform and organize them by attaching various tags. These tags are also used to create cross-references between bookmarks of different users. Flickr[6], existing since 2004, is a web site where users can upload and share their photos. Tags can be attached to all pictures and used to search for images.

Semantic tags are ambiguous. On the one hand they provide a simple and intuitive possibility to describe an item, with only very few or even no restrictions. Users can

[4] http://last.fm
[5] http://delicious.com/
[6] http://www.flickr.com/

utilize any term that comes to their mind as tag. But this freedom can also turn into the major disadvantage. Since it is possible to use any term as tag, their meaning is often hidden, or does not contain any useful information for the general user. A very popular example for this is the tag *seen live* attached to a song, which expresses that the user who tagged this song has seen a live performance of the artist or song. That may not give much information to others, though it might express the popularity of this artist.

Another problem arises through miss-spelled tags, which makes matching between tags more difficult. Most of these problems are evaded if there is huge amount of tags, where the majority of the tags will be correct or, if not correct, probably used by the majority of the users. This works for popular songs which have up to several hundred tags attached, but leaves out new, alternative songs with only a small number of tags.

In the field of music information retrieval tags are an upcoming research topic. In the last two years, the Music Information Retrieval Evaluation eXchange[7] [Dow08] (MIREX), an annual evaluation competition for music information retrieval algorithms, has launched tag-related tasks in the annual competition where the challenge is to automatically attach tags from a predefined set to a given song [HBC09, LWM+09]. On the other hand, Laurier et al. used tags to determine the mood of a given piece of music [LSSH09].

### 2.2.3 Expert-based Cataloging

Catalog based approaches are inspired by classical libraries, where librarians read and categorize the individual documents. For audio collections this means that experts listen to each track and then assign tags and, for example, genre labels or mood categories. These labels are often implemented as semantic tags or embedded metadata. The sheer amount of audio tracks available in today's audio collections makes this approach expensive compared to the previous listed approaches, but the technical requirements and complexity are rather low.

Allmusic[8] and Pandora[9] are two examples where audio tracks are classified and rated by human experts. Pandora provides an online music recommendation service together with online radio streams that are automatically generated, while Allmusic is a music catalog in the classical sense providing its content to end users and music stores.

The major drawback of this manual approach is that, apart from the high costs

[7] http://www.music-ir.org/mirexwiki/
[8] http://allmusic.com/
[9] http://www.pandora.com

for the human experts, many labels strongly depend on the personal experience and musical taste. A prominent example would be the *mood* of a piece of music, which strongly depends on the personal taste and even on the current emotional condition, but also labels that are more objective on the first glance such as *genre* can be difficult to decide, e.g. whether a song should be categorized as *alternative rock* or *indie rock*.

### 2.2.4 Collaborative Filtering

Collaborative filtering [GNOT92] is a technique that hast been used since long times, e.g. as recommendation system on Amazon[10], appearing as *"Customers Who Bought This Item Also Bought..."* on an articles detail page. In the field of electronic audio track organization and recommendation the technique has only recently arisen due to the fact that online music repositories are relatively young.

The essential is to make automatic predictions (filtering) about the interests of a single user by collecting and combining information from a large number of users (collaborative). The underlying assumption of this approach is that users who agreed in the past, tend to agree again in the future. A collaborative filtering or recommendation system for music tastes could make predictions about which music a user should like given a partial list of songs of that user's tastes (likes or dislikes). Last.fm is an online radio station that is using collaborative filtering together with semantic tags to provide personalized music to its users. In 2008, Apple introduced iTunes Genius which also relies on collaborative filtering based on the users personal audio collection and recommends new songs from the iTunes store.

Netflix[11], an online subscription rental service for movies, uses collaborative filtering to recommend movies to their customers. In 2006, Netflix launched the *Netflix Prize*, a contest for a recommendation algorithms that caused a great echo in the media. The first algorithm returning 10% better prediction results than the current prediction systems used by Netflix would be rewarded with 1 Million US$[12]. The challenge was successfully completed in fall 2009.

Collaborative Filtering is approved and widely accepted by both, companies and customers. It has been used for almost a decade and a wide range of ready-to-use systems is available on the market, commercial products as well as open source solutions. But on the downside, collaborative filtering suffers the so called "cold start problem": music (or more generally "a product") that has been published recently is disregarded by the system unless quite a large group has added it to their collection

[10] `http://www.amazon.com`
[11] `http://www.netflix.com/`
[12] `http://www.netflixprize.com/`

and listened to it. Consequently, users will be presented music mainly from the pool of very common and frequently listened music, while new or less popular music is underrepresented and has difficulties to gain importance.

### 2.2.5 Map-Based Music Organization

As a basis for the following applications and scenarios *Music Maps* [RF01] are used to create an interactive and playful interface to large audio collection. *Music Maps* provide a special way of organizing music collections by presenting a music library as a landscape where certain musical styles are automatically grouped together on islands.

A *Music Map* is created by a Self-Organizing Map (SOM) which arranges the music on a rectangular grid in such a way that music that sounds similar is located close together. A SOM is an unsupervised learning algorithm that projects data point from a high dimensional input space (feature space) into to a low dimensional output space (map space), commonly a 2-dimensional grid, while preserving the topology [Koh95]. The high dimensional data used as input are feature vectors extracted from the audio files as described in Section 2.1.

In detail, the process to create such a SOM is as follows: After analyzing the audio files, the high dimensional feature vectors are provided as input to the SOM training algorithm. The map consist of a user-definable number of units, which are usually arranged on a 2-dimensional grid. The training process works as follows:

1. Each of the units $i$ is initialized with a model vector $m_i \in \Re^n$ (also called weight- or prototype vector) of the same dimensionality $n$ as the feature vectors. In most cases, the initialization is done randomly, but other methods are possible. For more details about SOM-initialization methods please refer to [ABA05].

2. The following steps are repeated until the stop-criteria (e.g. the number of iterations $T$) are met.

   a) At each iteration $t$, a randomly selected feature vector $x$ is compared with the all model vectors to find the closest unit (the winner or best-matching unit (BMU)) $c$:

   $$c : ||x(t) - m_c(t)|| = \min_i \{||x(t) - m_i(t)||\}, \tag{2.1}$$

   where $||\cdot||$ denotes the distance function. The distance function is defined by the chosen metric (e.g. euclidean- or city-block-metric) and expresses the reverse "similarity" between two vectors in the input space.

b) An adaption of the model vectors is performed as follows:

$$m_i(t+1) = m_i(t) + \alpha(t)h_{ci}[x(t) - m_i(t)], \tag{2.2}$$

where $\alpha(t)$ is the lernrate, a monotonically decreasing function. For the neighborhood function $h_{ci}(t)$ different functions can be applied, the most prominent is the gaussian

$$h_{ci}(t) = e^{-\frac{||r_i - r_c||^2}{2\sigma^2(t)}} \tag{2.3}$$

where $||r_i - r_c||$ is the distance between the units $i$ and $c$ in the output space, and $\sigma(t)$ is the width of the neighborhood kernel, decreasing over time.

The effect is that the winner unit $c$ is "moved closer to" the selected feature vector $x$, and so are the units in the neighborhood, but to a lesser degree than the winner (depending on the neighborhood function $h_{ci}$). This enables a spatial arrangement of the feature vectors such that similar vectors are mapped onto regions close to each other in the grid of the units.

Since the learnrate $\alpha(t)$ and the width of the neighborhood kernel $\sigma(t)$ are functions decreasing over the time $t$, the number of units that are adapted is steadily narrowed and the degree of adaption is reduced.

3. After the training phase, an optional fine-tuning phase can be added. All feature vectors are once more matched with their winners, but in this step only the winner unit is adapted with very low lernrate. [GCM08]

4. Finally, each feature vector is mapped to its best-matching unit on the map.

There are various possibilities of stop criteria in Step 2. Limiting the number of iterations is often used, and one of the most simplest to implement. Other criteria can be to stop the training process when the map has reached a predefined quality measurement or the the SOM has converged to a stable state, and thus changes within the last $z$ iterations were below a threshold $\psi$.

A SOM trained with a set of audio features, typically *Rhythm Patterns*, creates a *Music Map*. On this map, music that sounds similar is located closely together in groups or clusters, visualized as islands on the map. On the other side music of different style is separated. Tracks that do not fit clearly to an island are placed in between, creating smooth transitions from one musical style or genre to an other [RPM03, NDR05]. PlaySOM and the PocketSOM applications utilize map-based music organization for large audio collections on desktops and mobile devices. Prototypes of both are presented in Section 2.5.1 and Section 2.5.3 respective, Chapter 4 presents an enhanced version of PocketSOM with several additional collaborative features.

## 2.3 Social Music Consumption

In many cases, music is consumed collectively, since a group of people listening to music is gathered rather quickly. Selecting music for different groups is, however, a difficult task. In rather small groups where the members know each other this is probably solved by simple discussion. The more challenging question is how to collaboratively select music and create playlists in larger, more heterogeneous groups where the individuals do not know each other personally.

Different approaches have been developed to address these questions. In [OLJ+04] O'Hara et al. propose a system for public places such as e.g. a bar, that allows the audience to vote for the forthcoming song. While a song is running, people can nominate songs to be played. The nominated candidates are afterwards put on a public election where the visitors of the bar can vote for the song they want to hear next.

The system completely relies on the result of the election and thus on the music competence of the audience. It is not guaranteed that two songs in sequence do fit to each other unless the majority of the voters do know the candidate songs. Furthermore, the system tends to repeat certain songs throughout the day, since most people tend to vote for the familiar ones. On the other hand, the playlist can be "hijacked" by a coordinated minority changing the played music to a completely other style.

Leitich and Toth [LT07] propose a system where the audience as well nominates candidate songs to be played next. The system then analyzes the songs nominated and determines the best matching song with respect to predefined criteria and the song being currently played. The criteria for the final song selection can be based on extracted audio features (c.f. Section 2.1), for example the song played next is the one with the smallest distance to the current song, or on meta-data such as id3-tags, e.g. the song with the same genre-label, but from a different artist and the closest year of creation. In contrast to the previous system, since the final decision is based on objective similarity, it can more likely prevent immediate changes in the musical style.

## 2.4 Automated Playlist Generation

DJs select tracks which blend together best, and a radio station covers a certain style to attract and hold listeners. An automated playlist generator is supposed to do this on a given audio collection, thus acting as a personal DJ. Being a good DJ, i.e. creating good playlists, is difficult. With the growth of audio collections, the creation of playlists gets more and more difficult, since the range of possible candidate songs for the playlists increases.

Automated playlist generation relies on a data structure to be able to select songs for a playlist. Different types and modalities of data can be used, such as meta-data, user generated tags, social networks or content based descriptors to create such structures. The quality of the generated playlist strongly depends on the type, the availability and the quality of this data.

Depending on the data source, there are various ways how the user can explicitly or implicitly control the playlist generation process. The simplest way is to define various *meta-data based constraints* (e.g. "only songs from the rock-genre", or "not more than one song per artist") that are entered explicitly by the user.

Other generation systems accept a *seed song* defining the starting point of the playlist. Last.fm, iTunes Genius and Pandora are popular examples of applications using seed songs to automatically generate a playlist. The user picks an arbitrary song and the system will provide the subsequent tracks, selected from its repository by e.g. collaborative filtering (last.fm, iTunes, c.f. Section 2.2.4).

In most cases, automatic playlist generation is transferred into an optimization problem. The optimal playlist is expressed by a goal function. The requirements posed by the user are implemented either into the goal function or added as constraints to the optimization problem. Various strategies to solve the optimization problem are used in the different approaches:

In [Log02], Logan uses simple *heuristics* to create a playlist. The audio collection creates a graph where each song is a node and links between the songs describe how closely the songs are related. Logan uses spectral features to calculate the link-strength. Starting from a user defined seed song, the system creates a playlist by finding the shortest path of the desired length in the graph. Additionally, Logan uses a simple version of *automatic relevance feedback* to improve the generated playlists: the first song is the song with the shortest distance to the seed song. The second song is the song with the shortest path to both, the first song and the seed song. In general, to find the next song for the playlist, the $N$ previous songs of the playlist are taken into account.

Pauws et al. [PVV06, PVV08] use *local search* combined with *simulated annealing*, an optimization heuristic inspired from annealing in metallurgy, to generate playlists. A playlist is described as optimization problem that is to be solved. Songs are described by several high- and low-level attributes and features such as artist, title, duration or tempo. The model of Pauws et al. defines several global and local constraint types that can be used to configure the playlist. The target of the optimization problem is to find a playlist with the highest similarity measure satisfying the predefined constraints.

Alghoniemy and Tewfik [AT01] formulate the process of creating a playlist in a *network flow*. A playlist is defined as a flow in a network from source to sink. The

capacity of the links between the nodes (songs) in the network are weighted according to a user defined similarity measure. The properties of the playlist can be defined through constraints. For network flows, source and sink (first and last song in the playlist) have to be known, thus they have to be defined by the user. To overcome this restriction, Alghoniemy and Tewfik define a dummy-source and a dummy-sink to keep the network flow model. These dummy-nodes are not considered in the final list. The playlist is created by finding the optimal flow in the network under the defined constraints.

Neumayer et al. [NDR05, NFH+07] use *Music Maps* to generate playlists. The technical and algorithmic background is briefly described in Section 2.2.5. Playlists can be created by selecting regions on the map or along a trajectory on the map. The possibilities of user interaction are manifold: The user can select a region or draw a trajectory, pick points on the map which are then interpolated to a trajectory, or define seed-songs. A prototype of such a system is described in Section 2.5.1.

Flexer et al. [FSGW08] use Gaussian models in conjunction with the Kullback-Leibler divergence to model a playlist as optimal steps between given start- and end-point. For each song in the database, MFCC features are computed (c.f. Section 2.1.5) and a single Gaussian model is trained. Then a similarity matrix is created using the Kullback-Leibler divergence, a measure of the difference between two probability distributions, and the respective Gaussian models. The playlist is generated by filling the gap between the start and the end song with based on the similarity matrix – having increasing divergence to the start song and decreasing divergence to the end song as the playlists progresses. The concept of Flexer et al. been used as an alternative interface and recommendation system for the FM4 soundpark[13] [GF09].

## The Future

In the last years, a new trend appeared for high quality (and high prize) products to adapt their settings autonomous using input retrieved from built-in sensors. Such ”ambient sensors“ have been used for several years in luxury class cars, where e.g. the headlights are automatically turned on in the dark or the windscreen wipers are turned on when it starts raining, but this trend is now emerging to other fields, especially to consumer electronics. A wide spread example are LCD monitors or laptop screens adapting their brightness to the intensity of the environmental brightness. In the sunlight or in a bright light office, the screen brightness is turned up, in the evening when the lights are dimmed also the brightness of the screen is turned down to save the user's eyes.

[13] `http://fm4.orf.at/soundpark/`

In the field of automated playlist generation, ambient and implicit parameters derived from the user's environment are currently not used in any system, but it is only a matter of time until first systems will be presented on the market. The possibilities for ambient and implicit parameters are virtually unlimited: Mouse movement and keystrokes can be used to determine the current activity level of the user to adjust the playlist and play faster music during high activity levels and slower music in periods of low activity. Equipped with additional sensors or attached input devices, the system can use current information from the environment: The number of people in the room (from the webcam) and the current sound level (from the headset) can give a hint whether there is currently a party (thus inciting music is needed), or in combination with the lighting conditions (again from the webcam or from the LCD monitor mentioned above) the music is for a romantic candlelight dinner.

These ubiquitous approaches of interacting with an audio player can possibly ease the use of home audio collections. On the other hand, such systems must be well trusted by the end users, and of course configurable: not everyone likes being watched by a computer...

## 2.5 Applications

This section will briefly present selected applications that use *Music Maps*, or a similar metaphor, to provide access to a digital audio library.

### 2.5.1 PlaySOM

PlaySOM is an application that provides interactive and intuitive access to large audio collections based on *Music Maps* as described in Section 2.2.5. The main window of PlaySOM (see Figure 2.4) consists of the *Music Map* and allows the user to create playlists by drawing on it. Several modes of selection are available: By rectangular selection, entire clusters of music containing similar sounding tracks are added to the playlist. By drawing trajectories, it is possible to create a playlist going from one musical style smoothly to one or various others, according to the path selected. The internal player module shows the songs of the current playlist where the user can further refine the playlist and, of course, play it. Figure 2.4 shows the main screen of PlaySOM with an example *Music Map*. PlaySOM further implements various visualizations [LR08] which help users to orient themselves on the map and aid in finding the desired music.

*Music Maps* can be exported from PlaySOM to various other formats, e.g. to a HTML website or as Flash application. Also PocketSOM uses a simplified export

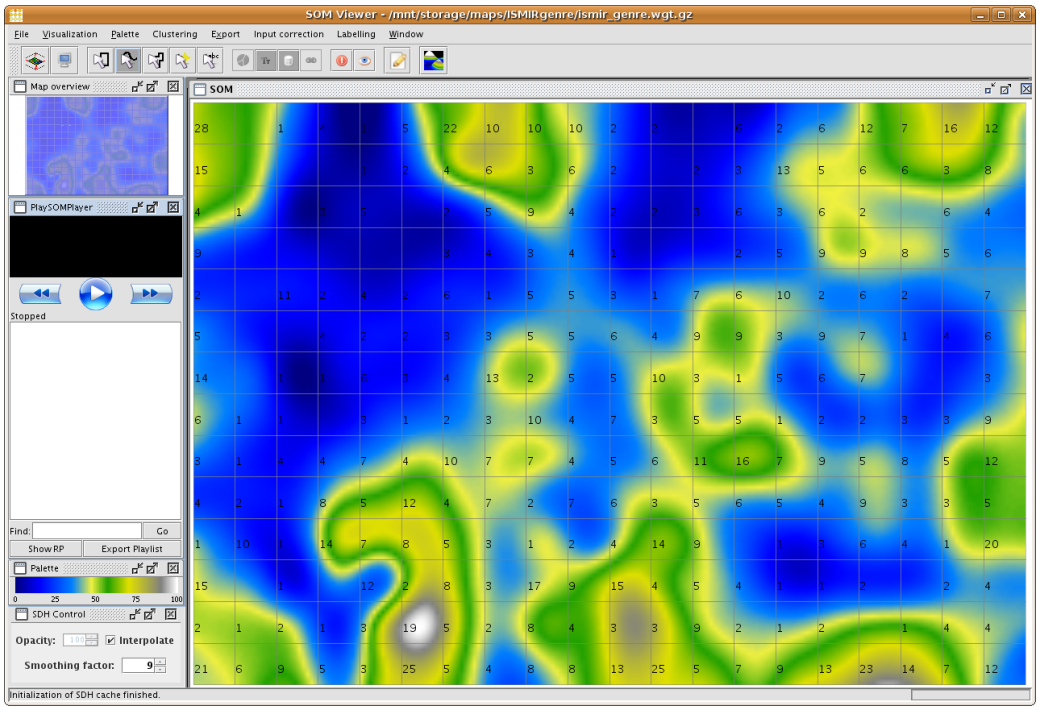



Figure 2.4: The PlaySOM application showing the ISMIR genre *Music Map*

format of *Music Maps*– either file based or through a direct connection to PlaySOM.

**HTML Export** The first online demo of PlaySOM and *Music Maps* was enabled through a simple HTML-representation. The user can explore the map by selecting the individual units and listen to the audio files placed on them. For a better understanding of the map, the user can switch between several, predefined visualizations. A demo installation based on the *Map of Mozart* [MLR06] is available online[14].

**PlaySomFx** A more advanced online version of PlaySOM is PlaySomFx, a Flash based implementation to interact with *Music Maps*. The user can not only explore the map and zoom in to get a more detailed view, it is also possible to create playlists based on rectangular and trajectory selection which can be played using a local audio player. Again, the user can switch between different predefined visualizations. A demo installation is available online[15].

### 2.5.2 MusicMiner

MusicMiner[16] is another application visualizing audio collections as a music map. Songs are placed on the map according to their audio similarity thus creating clusters of songs representing a certain style of music [MUTL06]. The audio similarity is based on MFCC and temporal higher level features [MUT+05], the map is based on an Emergent SOM (ESOM [UM05]), a large SOM with several thousand units.

The Music Miner provides a topographical map of a music collection and allows to play songs directly off the map, or to create playlists from regions of paths on the map.

### 2.5.3 PocketSOM

PocketSOM is a simplified viewer for *Music Maps* on mobile devices. It was developed in 2005 [NDR05] to bring the advantages of map-based interfaces to portable devices (see Figure 2.5(a)).

The usage of PocketSOM is simple and intuitive:

- After loading the data file, containing the map and the references to the music files in the collection, the *Music Map* is displayed.

[14] `http://www.ifs.tuwien.ac.at/mir/mozart/`
[15] `http://www.ifs.tuwien.ac.at/mir/playsomfxweb/demo/PlaySomFx.swf`
[16] `http://musicminer.sourceforge.net/`

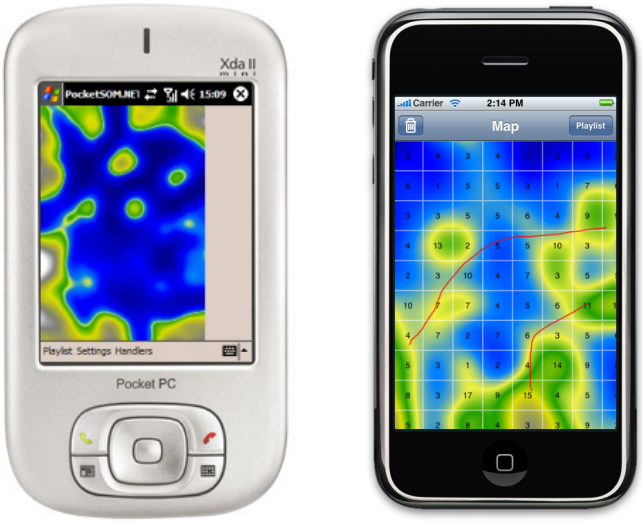

(a) iPocketSOM (b) mPocketSOM

Figure 2.5: The *PocketSOM Family* on selected devices.

- Using the touchscreen, the user can now draw a path on the map which will result in a playlist. The songs on this playlist are selected along the trajectory drawn.
- Afterwards, the user can manually edit the playlist, e.g. deleting or reordering items.
- Finally, the playlist is sent to an audio player. Playing from local storage as well as streaming songs remotely is supported.

PocketSOM is based on Java and Eclipse SWT (Standard Widget Toolkit), and was developed and tested on an **iPAQ Pocket PC** which comes with the *JeodeVM*, a special Java-VM for the iPAQ. *iPocketSOM* can be run on any device that provides or can install a standard Java version 1.2 VM.

Since this first prototype application, *PocketSOM* has been implemented for different devices based on different platforms. The reason for these many different implementations is quite simple. Since applications created for mobile devices are subject to certain device specific restrictions, it is almost impossible to provide one application appropriate for all different kinds of platforms available. To avoid the "one-size-fits-none" problem several different and independent implementations, each based on a different platform, have been developed. While each implementation makes use




(a) PocketSOM.NET (b) iSOM

Figure 2.6: The *PocketSOM Family* on selected devices.

of special features provided by the platform it is designed for, all share the common core-set of main features listed above.

The main interactions with *PocketSOM* such as drawing a path on the *Music Map* or editing the playlist are done via the touchscreen. Textual input which would require to use the keyboard (e.g. a server address) is either kept to a minimum or assistance will be provided, e.g. through a file open dialog and is saved as long as the application is running so that these settings have to be set only once.

In the following sections the different implementations of *PocketSOM* will be presented. *ePocketSOM*, which is part of this thesis, will be presented in Chapter 4. The "original" *PocketSOM* and its variants together form the *PocketSOM Family*.

### PocketSOM for J2ME

Figure 2.5(b) shows *mPocketSOM* [Hla07], an implementation of PocketSOM application based on JavaME[17], the Java Micro Edition (formerly J2ME). JavaME is supported by many portable devices such as PDAs or mobile phones, thus *mPocketSOM* will run on any mobile device that provides a touch screen, the Connected Limited Device Configuration (CLDC) 1.1, and the Mobile Information Device Profile (MIDP)

[17] `http://java.sun.com/javame/`

2.0.[18] Since most *Music Maps* use MP3 audio files, *mPocketSOM* additionally requires an implementation of the Mobile Media API supporting MP3 playback[18]. The application was tested on different mobile phones, including **Nokia 7710** and **Sony-Ericsson M600**.

*mPocketSOM* also supports zooming to provide for more immersive user interaction even on mobile devices, but many devices have, however, technical difficulties with this feature because of memory allocation problems, resulting in a program crash or system freeze. Therefore it is omitted in the final release, but is still an option for future implementations.

Additionally, *mPocketSOM* can act as remote control using a simple applet called XPlayer. XPlayer, which is part of the *mPocketSOM*-package, runs on a remote host. It waits for incoming requests and forwards them to an arbitrary audio player available on the remote host.

### PocketSOM on the .NET Compact Framework

PocketSOM.NET, based on the .NET Compact Framework is shown in Figure 2.6(a). Since PocketSOM.NET is developed in a native environment, its performance and integration in the operating system is very good. The .NET Compact Framework is available for Windows Mobile 2003 and Windows Mobile 5 and comes preinstalled with Windows Mobile 6. PocketSOM.NET was developed on the **HTC Magican** and tested on **BenQ P50** and the **MDA Compact II**. Since Windows Mobile based devices are gaining popularity, PocketSOM.NET will soon support a wide range of devices.

PocketSOM.NET supports loading a map over an Internet connection and local playback as well as sending the playlist to a remote VLC server. Songs played locally can be streamed from a remote location or taken from local storage.

### PocketSOM for the iPhone

iSom is the most recent implementation of *PocketSOM* (Figure 2.6(b)). It is a native iPhone application that uses the iPhone SDK and integrates perfectly into the environment by offering the well known look and feel.

The *Music Map* can be navigated using simple finger gestures. Drawing paths results in a play-list being generated which can be fully edited with simple tap and drag actions. Panning the map is done by simply touching the display and dragging the map around. By pinching two fingers together or apart, the zoom level of the map can be increased or decreased.

[18] for details see `http://jcp.org/en/jsr/`

A map is loaded from a web server via WiFi or cellular network. The music is streamed as well from a remote server, to allow access to audio collections that can be many times larger than the mobile phone's storage capacity. In addition, iSom can act as a client for the PocketSOM Connector presented in Section 4.1.1, allowing the mobile device to remotely control the internal player of the *PlaySOM* application and to share playlists and paths with the server and other clients connected.

## 2.6 Summary

This chapter gave an overview on related work and fundamental applications used for this thesis.

Different audio features, ranging from very basic low-level features to high-level musical concepts, have been briefly described. Further, several feature extraction systems providing implementations of various feature extraction algorithms were listed.

When the size of a repository of digital audio music exceeds a certain size, finding and selecting songs by their filenames gets very difficult. Various strategies exist to organize large music libraries, a set of today's most prominent approaches were discussed together with their advantages and disadvantages.

Various approaches and services for automatic playlist generation were described, ranging from theoretical concepts like a network flow model to available and heavily used services like last.fm.

Finally, the applications this thesis enhances and builds on were presented. PlaySOM, the application is used to create and view *Music Maps*, and PocketSOM transfers *Music Maps* to mobile devices,

The *Music Maps* used in the subsequent chapters are created by the PlaySOM application, using different combinations of the *Rhythm Histograms*, *Statistical Spectrum Descriptors* and *Rhythm Patterns* feature sets.

# 3 Creating and Manipulating Playlists

In this chapter five new approaches for creating and combining playlists based on the path metaphor are introduced. Combining playlists is a core functionality required to allow collaborative playlist creation.

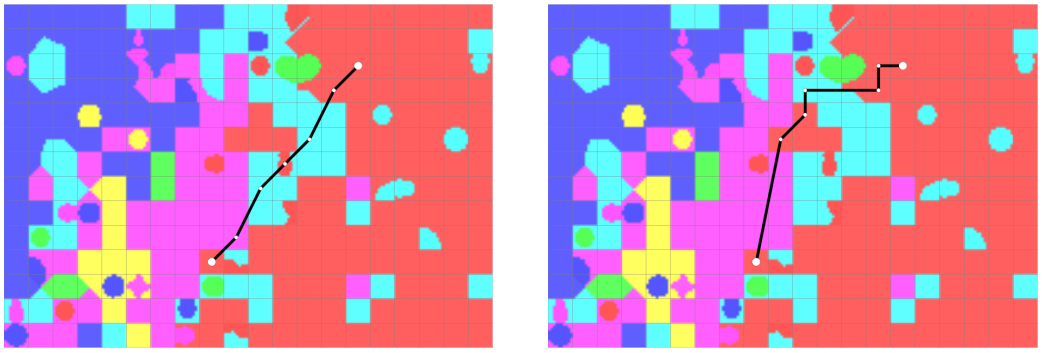
In the scenario, used in this thesis, users send their playlists (represented as paths) to a central *PlaySOM* application where all collected paths are merged into a single playlist. The combination is done manually at the *PlaySOM* application by a user who selects the playlists to be combined. The automation of this process is very complex, primary because the selection of the two playlists to be combined is difficult and the criteria to select the candidates need to be defined. Finding and defining these criteria and the limiting constraints will be part of future work.

In all of the approaches presented in this chapter, some kind of distance measurement is used during the combination process. This distance can be calculated in different spaces, such as the map (output) space or the input (feature) space (c.f. Section 3.1.1).

Generally, creating a playlist based on a *Music Map* is done by drawing a trajectory on the map. The playlist is filled with the songs that are mapped onto the units covered by the trajectory. In cases where more than one song is mapped to the same unit, the system can either add all songs, or select a predefined number of songs, either randomly or in a defined ordering.

## 3.1 Autorouting

Usually, a playlist on a *Music Map* is created by drawing a trajectory on the map. Here, a new method to create a playlist based on a series of landmarks, i.e. songs or positions on the map, is introduced. A playlist based on autorouting between landmarks is created by 1) (optionally) reordering the landmarks and 2) concatenating them, inserting intermediate positions between the landmarks to provide decent and convenient transitions.



(a) map-space routing (b) feature-space routing

Figure 3.1: Comparison of autorouted paths based on the same landmarks (end points) but computed in different spaces.

**Reordering** Before connecting the landmarks to a continuous path, the system can reorder them to optimize the covered distance of the resulting playlist. Currently a simple Traveling Salesman Problem solving algorithm is used, rearranging the landmarks in such way that the overall distance covered by the routed playlist will be minimal. Thus it can be avoided that the playlist keeps "jumping" from one corner of the map to the opposite because the landmarks where selected in random order.

**Intermediate Points** To provide more convenient transitions between the selected landmarks, the routing algorithm can insert a user-definable number of intermediate points, thus adding additional tracks to the playlist. While this expands and lengthens the playlist, the steps between the individual tracks get smaller, thus creating a more convenient transition from one landmark (over several songs) to the next.

The number of intermediate points inserted between two consecutive landmarks is currently statically defined for the whole playlist, while more sophisticated approaches of inserting additional songs are possible. Alternative methods are, e.g. to interactively insert a song between those those two points with the largest step width, or to distribute the available additional songs in a way to minimize the standard deviation of all steps to the average step width, i.e. keeping the steps between the songs at approximately equal length.

### 3.1.1 Routings

Calculation of distances between landmarks, which is needed in both of the previously described manipulations, can be based either in the high-dimensional input space (the

feature space) or the output space (the maps space). While calculations in the output space are more transparent to the end user, calculating in the input space can better reflect the original properties of the songs.

- **Map Routing**
  In this case, the routing is based in the output space. The shortest (and therefore preferred) connection between two landmarks on the map is the direct connection, a straight line. Distances for the intermediate points or the reordering can be directly read from the map.

- **Input Space Routing**
  Calculating the distances between landmarks in the input space in most cases leads to completely different results than in the previous approach and connections between landmarks seem to meander on the map.

  The calculation is done as follows. For landmarks specified by a song the corresponding feature vector is used directly, while for landmarks defined through a position on the map the model vector of the corresponding unit is used. Then, the calculation in the input space, the trajectory is transferred back into the output space by finding the best-matching unit on the map for the result vector.

  Input space routing can evade clusters of a complete different musical style that are located between two landmarks. The algorithm will choose a detour around the cluster to create a smooth and convenient transition.

A comparison of the input space routing and the map routing is shown in Figure 3.1. The map contains music from six genres – *classic* (red), *electronic* (blue), *jazz/blues* (green), *metal* (yellow), *rock/pop* (magenta) and *world* (cyan) – and is colored according to the musical genre that is dominant on each unit.

Based on the same landmarks as input, the routing algorithm inserted five intermediate points. The genres covered in the result are listed in Table 3.1. In Figure 3.1(a), the playlist, created through map routing, uses a direct connection between the landmarks, with only small fluctuations by snapping the intermediate points to the center of the units. The result is a varying playlist switching from classical music directly to rock/pop music advancing to world music.

The playlist in Figure 3.1(b) was created using the input space routing. It is clearly visible how the path avoids the area filled with rock/pop music and most of the world music between the landmarks and takes a detour over a small island of classic music.

The playlists created by both paths is shown in Table 3.1. While the playlist created by the map routing is more heterogeneous and covers more different musical styles, the input space routing almost constantly stays in one genre with only one sidestep

Table 3.1: Genres covered in the playlists created by Autorouting

| Song | Map Routing | Input Space Routing |
|---|---|---|
| Start | classical | classical |
| | rock/pop | classical |
| | world | classical |
| | classical | world |
| | world | classical |
| | classical | classical |
| End | classical | classical |

to an other genre. On the one hand, this results in better, more smooth, transitions between the songs for the latter playlist, whereas on the other hand the map routed playlist contains more variety and thus is more interesting.

## 3.2 Concatenation

When a group of people enjoys music, interests of different individuals have to be unified to provide a good experience for everyone. Novel concepts to collaboratively create a playlists based on contributions from multiple users are introduced in the next sections. The general approach is that the users create their individual playlists which are then combined by merging and/or concatenating them into one joint playlist.

The most straight forward way to combine playlists is to play just one after the other. In the metaphor of the *Music Map*, this means to connect the end of the first playlist with the beginning of the second. The sequence of the playlists (which one is the first, which the second) can be based on different properties, such as creation time, length, position on the map and others. Other possibilities are to automatically rearrange the playlists in a way that the gaps that have to be covered are minimized, or to just let the user explicitly define the sequence.

In some cases, the gap between two concatenated playlists will be too large, thus breaking the atmosphere that was created by the former playlist. In such cases, this gap can be filled with intermediate songs based on the auto-routing algorithm presented in Section 3.1. Figure 3.2 shows an example to two playlists ($P = \{p_1 \dots p_l\}$ and $Q = \{q_1 \dots q_m\}$) that are concatenated with five intermediate songs filling the gap in between.

The advantage of this approach to combine two or more playlists is that the technique is rather simple and easy to understand, even for non-sophisticated users, and thus improves the acceptance among users. The major drawback is that users who's

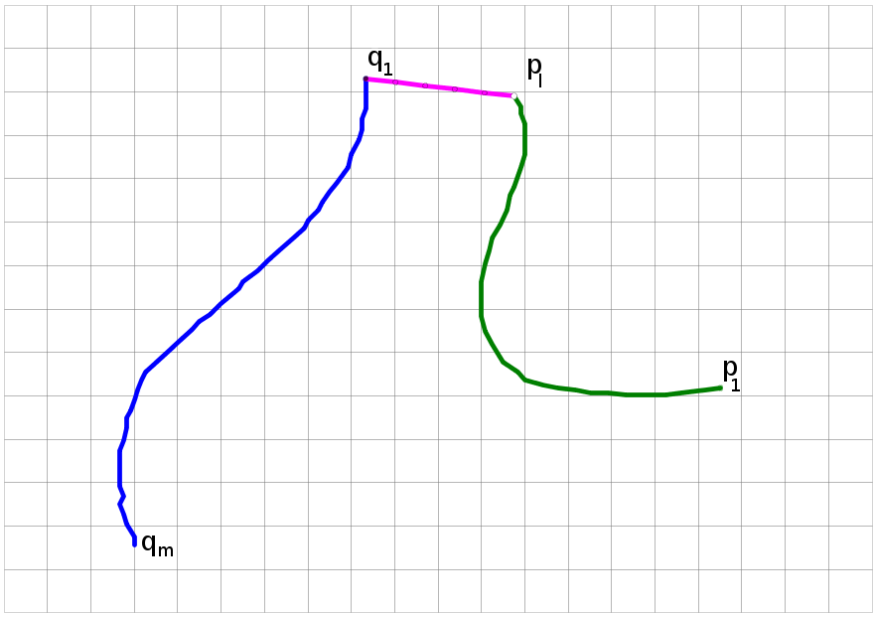



Figure 3.2: Path concatenation with intermediate steps.

playlists are scheduled towards the end have to wait a long time until songs they selected will be played. One possibility to improve this situation is by limiting each playlists to a certain length to reduce the total length of the playlist, or by playing only every $x^{th}$ song on the combined playlist, and after finishing returning to the beginning of the path, thus creating a loop cycling the playlists until all songs on the playlist have been played once.

Combining playlists by concatenation is useful when the gap between end and start is rather small, the distance between two playlists is too large to use an intermediate path, or if two playlists follow a more or less parallel path but in opposite directions.

## 3.3 Intermediate Path

For two playlists that follow a more or less parallel path on the *Music Map*, and have the same orientation, an intermediate path as a combination of both is a possible solution. The idea is to find a new path lying between the two baseline paths, keeping equidistance to both. This results in a new playlists that, also in the musical style, lies in the middle of the baseline playlists. This implies that songs that were selected by the users will possibly not be part of the new playlist. In most cases, when the baseline-playlist was created by drawing a trajectory on the map, users won't notice

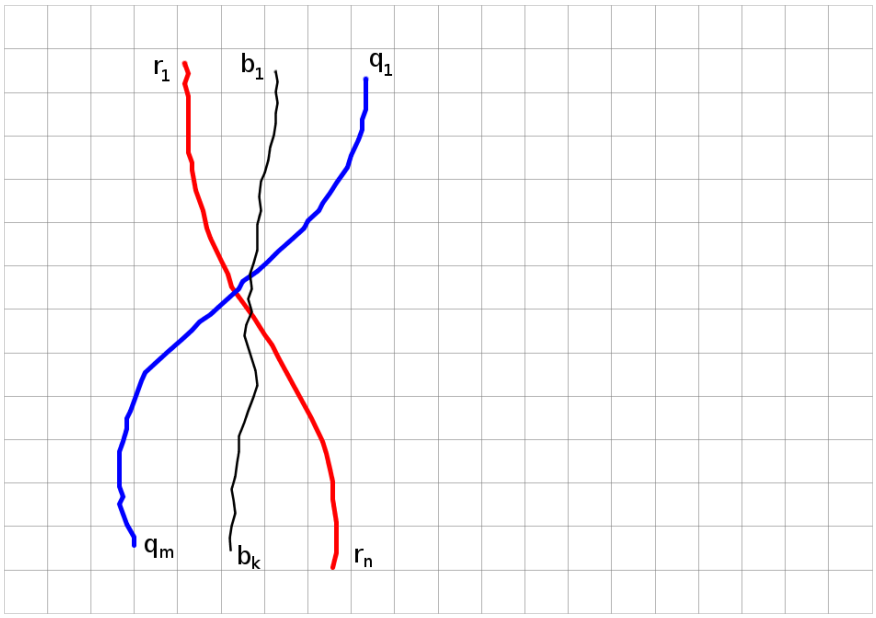



Figure 3.3: Intermediate path based on the map distance measure

the difference. On the other hand, when the path playlist was created by explicitly chosen songs, these songs should be included with a higher priority – a task that is currently placed on the user supervising the merging process.

If the playlists are rather similar, the advantage of this approach is that the resulting playlist will fit for both playlists that were used as input, thus giving two users at the same time "their" playlist. On the other hand, when the distance between the input-playlists too big, the new playlists might cover a complete different musical style. Depending on the size, resolution and the layout of the *Music Map*, it might be necessary to restrict this approach to paths from the same cluster.

An intermediate path $B = \{b_1...b_k\}, k = min\{m, n\}$ is created based on the two paths $Q = \{q_1...q_m\}$ and $R = \{r_1...r_n\}$. The algorithm chooses $b_i$ such that the distance to the respective points on both paths to be merged is equal:

$$||b_i - q_i|| = ||b_i - r_i|| \tag{3.1}$$

For a map based intermediate path, the calculation is based on the coordinates of the points $q_i$ and $r_i$ on the map. For a input space intermediate path, the points are transferred into the input space by using the model vector $m$ of the unit the point is located on. $m(q_i)$ denotes the model vector of the unit on the map where the point $q_i$ is located. The intermediate path is first calculated in the input space by creating

$C = \{c_1 ... c_k\}$ such that

$$||c_i - m(q_i)|| = ||c_i - m(r_i)|| \tag{3.2}$$

The path $C$ is then transferred back into the map space, creating the final intermediate path $B$, by finding the best matching unit (BMU) for each of the points $c_i$

$$b_i = pos(winner(c_i)) \tag{3.3}$$

where $winner(c_i)$ selects the BMU $u$ according to Equation 2.1 and $pos(u)$ selects the center-coordinates of unit $u$ on the map.

Figure 3.3 depicts the intermediate path $B$ (black) based on $Q$ (blue) and $R$ (red).

Analogous to the Autorouting, an Intermediate Path based on the input space creates a playlist that is musically closer to the original playlists but tends to break the visual representation on the map unless the *Music Map* is very fine grained and contains a high number of units. In most cases, a map based Intermediate Path creates a playlist with larger variations but visually keeps concept of an intermediate path which might be preferred by the user.

## 3.4 Sub-Path Recombination

Before applying any of the above methods, a path can be split into two or more sub paths, which are then combined with other paths. The split can be applied automatically, by cutting the path into two parts of equal length, or at a user-defined position. This is especially useful if two playlists that should be combined using an intermediate path (see Section 3.3) significantly differ in length.

As part of future work some advanced automatic splitting algorithms may be developed, such as splitting a path at the border of dominant clusters or at an intersection with another path.

These sub-paths allow much more fine-grained manipulation of paths and playlists which can improve the quality of the resulting playlist. On the other hand, a path split in too many segments will loose the characteristics of the original path and result in almost arbitrary playlists when recombined afterwards.

## 3.5 Multiple Combinations

Every manipulated path results in a new path on the map which can again be used as input together with other paths for another modification. This allows to create

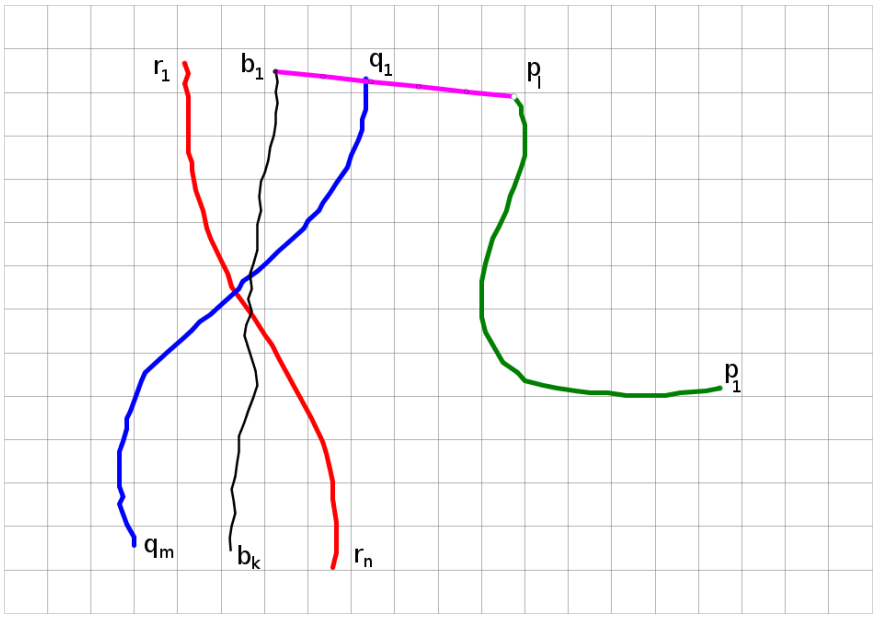



Figure 3.4: A combined path, created by an intermediate path and concatenation.

one universal playlist from many different playlists by applying different techniques subsequently to the paths.

Figure 3.4 shows the overall result of combining the three baseline paths $P$, $Q$ and $R$. First, $Q$ (blue) and $R$ (red) were used to create the intermediate path $B$ (black). The new path $B$ was then concatenated to $P$ (green) (connected by the magenta line), resulting in the final path $F = \{p_1 \dots p_l \dots b_1 \dots b_k\}$.

## 3.6 Summary

Combining playlists created by different users is one possibility to enable collaborative playlist creation. The methods proposed in this chapter combine playlists based on the path metaphor of playlists on *Music Maps*.

Creating a playlist based on given songs by connecting them and inserting additional tracks between songs of strongly differing styles is one possibility of generating a collaborative playlist, which is also transparent and understandable for the users. Instead of connecting singe tracks to a playlist, also a series of playlists can be connected one after the other, again inserting additional songs to support changes in the musical style. Two playlist can be merged together, forming a combined playlist comprising characteristics of both.

The automation of the process of combining playlists is complex. The criteria to find the right candidates for a combination are difficult and need yet to be defined in a machine-understandable way. Finding these criteria and the limiting constraints of the process will part of future work.

# 4 ePocketSOM


Figure 4.1: *ePocketSOM* on a BenQ showing a *Music Map*

User interfaces on mobile devices have to fulfill special requirements regarding interaction and information presentation. Input and interaction possibilities are heavily limited by the small screen size and a keyboard of tiny keys, often with multiply assigned functions and sometimes even not existent at all. On-screen-keyboards, touchscreens, character recognition or special keys can provide basic input capabilities, but these might still be inappropriate for certain application scenarios. On the other hand, the mobility and agility of mobile devices enables new input capabilities through acceleration sensors by e.g. shaking, tilting or turning.

One major issue is to adequately display and provide access to very large data collections such as a music library. Scrolling through directory structures or search result lists using these minimal interface capabilities is even worse than on desktop PCs. On the other hand the presence of a touchscreen dramatically enhances the ease and intuitiveness of a *Music Map*.

*PocketSOM* (see Figure 4.1) is a reduced version of the *PlaySOM* application (c.f. Section 2.5.1) providing a simple and intuitive interface. Its functionality focuses on

creating and handling playlists using *Music Maps* created by *PlaySOM*.

#### PocketSOM and PlaySOM - a team for music experience

The combination of ePocketSOM and PlaySOM (c.f. Section 2.5.1) enables advanced additional features of interaction with the map, and allows multi user access to a shared map.

In the most simple scenario, *PlaySOM* acts as the map data provider, which creates the layout and visualization of the map, together with the audio files populating the map. Both the map and the audio data can be downloaded or streamed from PocketSOM over HTTP. Furthermore, *PlaySOM* can collect paths from several clients and combine them to one global playlist. These features are provided by the PocketSOM-Connector, a module of the *PlaySOM* application (see Section 4.1.1).

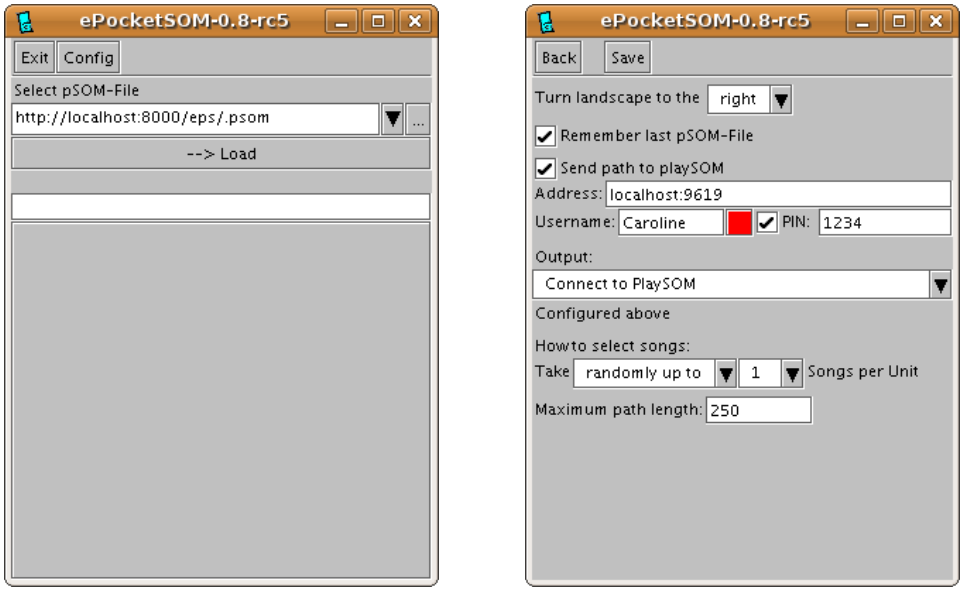
## 4.1 Implementation

*ePocketSOM* is an application for mobile devices offering interactive access to large audio collections, either from local storage or streamed from a remote server. It is based on Ewe[1], a Virtual Machine for portable devices supporting a large number of different devices and operating systems. Ewe strongly resembles Java and uses the same syntax, but replaces some of the default JRE libraries by specific implementations mastering the limited resources available on mobile devices. Applications developed in Ewe can be executed in the Ewe Virtual Machine, but also compiled and packaged to run native on all supported platforms thus simplifying the distribution and installation process.

Since the last major revision, Ewe has been renamed to Eve[2], and the list of supported devices was extended. Eve is available for devices running Windows Mobile 5/6, Windows PocketPC, the Nokia Internet Tablets N770 and N800, for Windows and Linux Desktops and any Java 2 Runtime Environment.

*ePocketSOM* consists of four main Parts: The *DataManagementPane*, the *LandscapePane*, the *PlaylistPane* and the *ConfigurationPane*. After start-up, the DataManagementPane (see Figure 4.2(a)) is shown where the user can select and load a *Music Map*. Maps can be loaded from local storage as well as from a remote web-server. The path or URL of the map is entered into the text box, local maps can be selected through a file dialog. Alternatively, the user can select one of the four most recently used maps, which are stored to allow quick access to frequently used maps.

[1] `http://www.ewesoft.com/M2.htm`
[2] `http://www.ewesoft.com/eve/index.html`




(a) The DataManagementPane (b) The ConfigurationPane

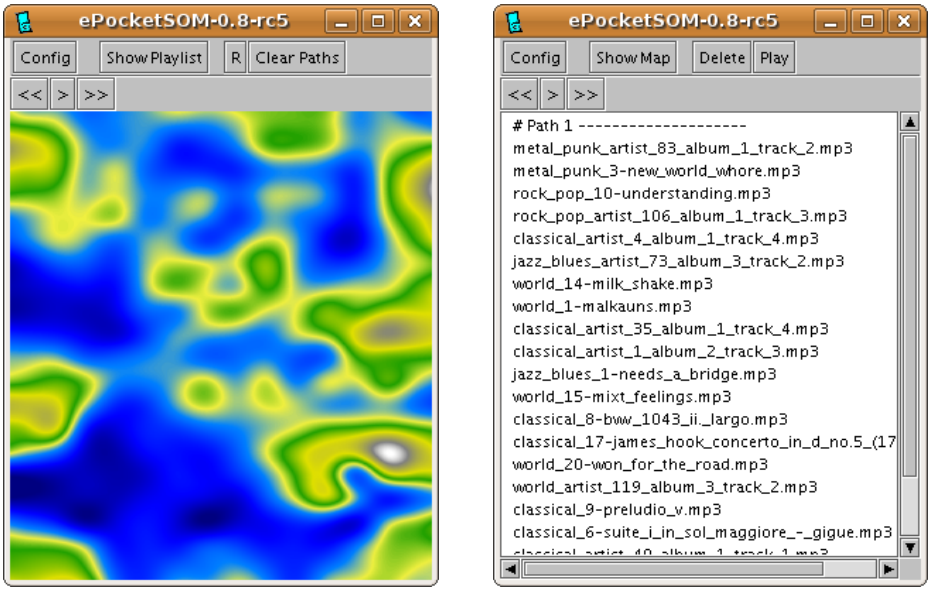
Figure 4.2: Selected Screens of *ePocketSOM*

After pressing the Load-button, the PSOM-file and all referenced files are loaded into the application. While the application is running, the DataManagementPane holds all available data such as the map-image or the mapping of songs on the map. After loading the *Music Map*, the LandscapePane is activated.

Three files are required to display a minimal map: A PSOM-file, a mapping file and a visualization image. The PSOM-file contains the basic configuration settings for the map, which is a reference to both the mapping file and the visualization image, and the common base-path (or base-URL) of the music files. Additionally the PSOM-file can contain the connection string of a PocketSOM Connector (see Section 4.1.1). If available, this setting is passed to the PlaySOM Connector Handler.

The mapping file contains a list of all songs of the map together with their position on the map. The list is preceded by three header lines specifying the size of the map and the maximum number of songs at one position. The visualization image can be provided as JPG or PNG and is scaled to fill the whole screen available. If necessary, the image is rotated from landscape to portrait layout (or vice-verse) to match the screen orientation.

The *LandscapePane* (see Figure 4.3(a)) displays the image of the *Music Map*. With the touchscreen the user can draw a trajectory on the map, thereby creating a playlist.




(a) The LandscapePane (b) The PlaylistPane

Figure 4.3: Selected Screens of *ePocketSOM*

It is possible to draw several independent paths on the landscape which are afterwards combined into a single playlist. After drawing a path the playlist is displayed in the PlaylistPane (Figure 4.3(b)). Here the user can further refine the playlist by deleting or reordering songs and then send the playlist to one of the registered Playlist Handlers (see Section 4.1.2). After the playlist has been sent to the Playlist Handler the LandscapePane is activated again to display the current playlist trajectory. The user can, however, freely switch between the Landscape- and the PlaylistPane.

On the *ConfigurationPane* (see Figure 4.2(b)), accessible from each other Pane, the user can select and configure the registered Playlist Handlers and define further parameters such as the selection mode or the maximum path length. The selection mode defines how the system chooses a song from a unit that should be added to the playlists – either randomly or in sequential order, while the path length limits the playlists to a given number of songs. It is further possible to configure the connection to a PocketSOM Connector (see Section 4.1.1) and activate related features such as sharing paths and playlists or enabling the remote control elements.

Based on Eve, *ePocketSOM* supports a wide range of different devices. It was developed and tested on a **BenQ P50 PDA phone** (running Windows Mobile 2003 SE), on a **MDA Compact II** running Windows Mobile 5 and on a **Nokia N770 Internet**

**Tablet**.

### 4.1.1 PocketSOM Connector

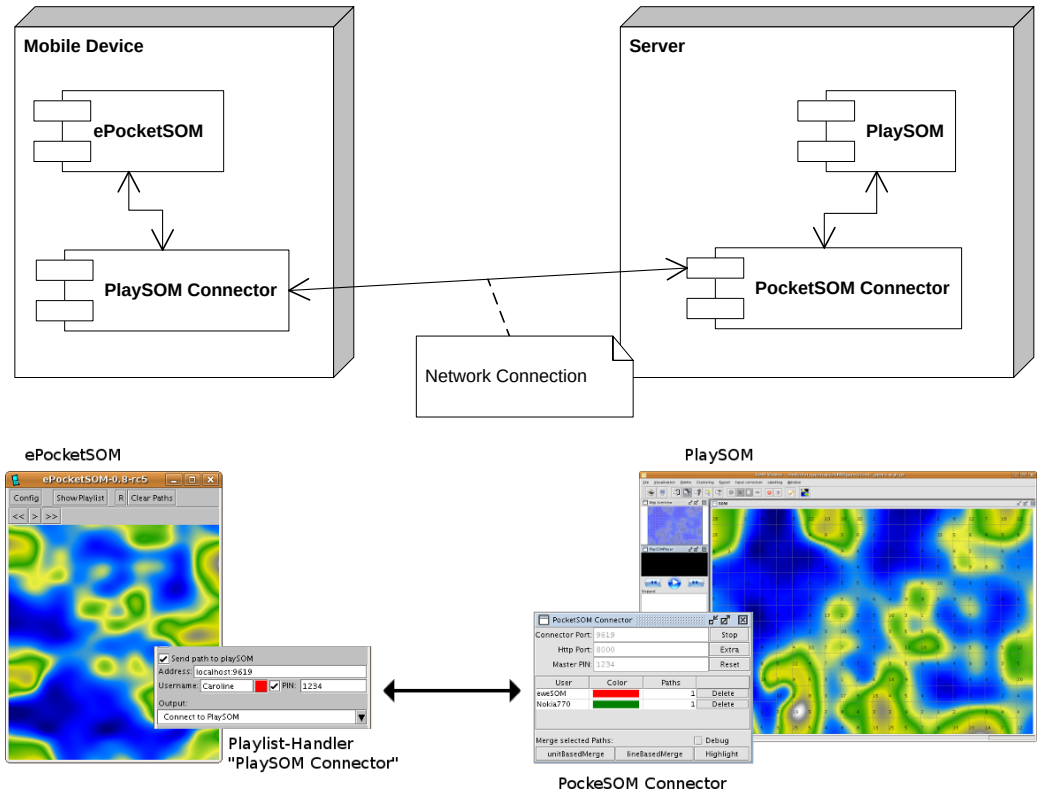



Figure 4.4: Overview of the architecture and components of the PocketSOM Connector/PlaySOM Connector

The PocketSOM Connector, part of the *PlaySOM* application (see Figure 4.4, right side), provides remote access to a *Music Map*. Four modules allow access to different parts and representations of the data included in a music map. The connector adds server functionality to PlaySOM.

The *Map Information Module* website provides basic information about the currently loaded map, accessible via HTTP through any web browser. The size of the map is shown together with the current visualization and the data files used, which are also available as download on the site (see Figure 4.5). Furthermore, it provides a basic overview of the distribution of the songs.

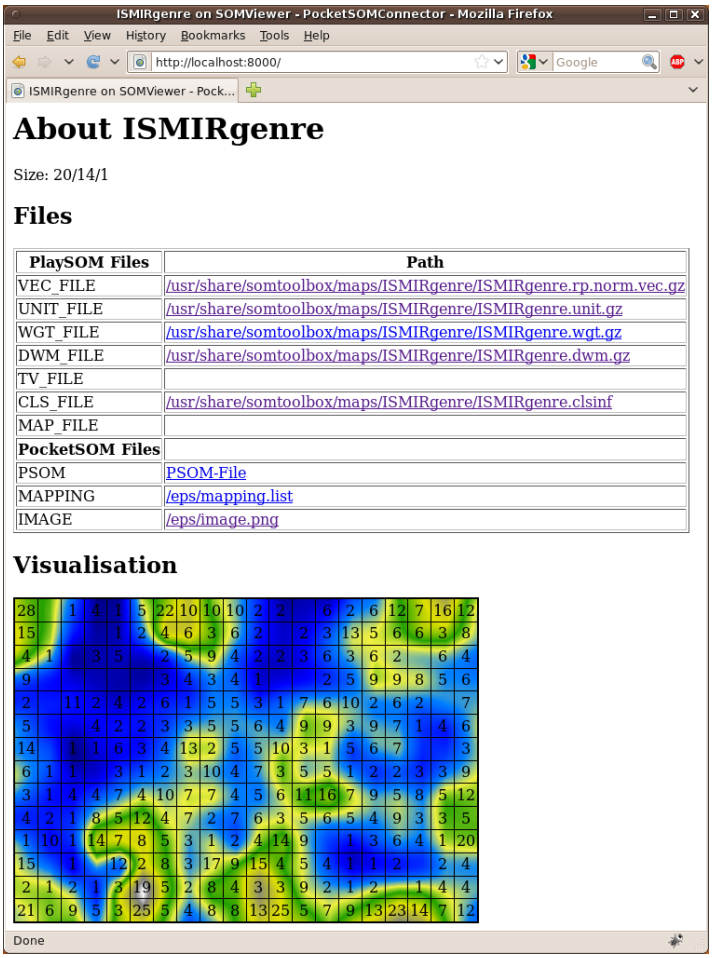



Figure 4.5: The Map Information Module of the PocketSOM Connector

The *Configuration Module* allows mobile clients running ePocketSOM to load *Music Maps* directly from the PlaySOM application by providing the required files via HTTP. The configuration files requested by the client are generated on the fly, which is especially useful for the visualization image which can change frequently. The *Music Provider Module* provides clients direct access the music files on the map, either as download or for streaming.

In combination with the PlaySOM Connector, described in Section 4.1.2, the *Interaction Module* allows some advanced interactive interaction schemes. The interaction module handles different communication streams between the PlaySOM (server) and the PocketSOM (client) application: for example, paths drawn on the map on a mobile client can be sent to the Interaction Module which displays them on the map of

the PlaySOM application. Some commands are only available after the client has authenticated itself to the server by sending the correct PIN. Clients can further register themselves to receive notifications from the server.

The following communication sequences are available at both, the PocketSOM Connector (server-side) and *ePocketSOM* (client-side), a detailed description of the protocol is available online[3]:

### Status Information

The client can request different status information from the server, such as the playlist of the internal player, the currently played song and it's position on the map.

The playlist is returned as a simple list of paths relative to the base-path defined in the PSOM file. The position of the current song on the map is returned in relative coordinates: `0/0` references the upper left corner of the map, `1/1` the lower right. `0.5/0.5` is the center of the map.

### Sending

Authenticated clients can send paths and playlists to the server, which are subsequently displayed on the map. The paths from each client (identified by a username) are grouped together and drawn in the same color. If the connection to the client is lost (or exceeds a timeout), all associated paths are removed from the map.

When a client sends a playlist to the server, the system behaves as if the playlist had been created locally. The playlist of the internal audio player of *PlaySOM* is filled with the songs received, and a notification is sent out to the registered clients. After that, the new playlist can be retrieved by requesting the playlist status.

### Notifications

The PlaySOM Connector will inform all clients that have registered themselves about events that occur in the PlaySOM application by sending a notification message. The following notifications are available:

- `currentsong <song-id>`
  This message is sent to remote clients when the internal player of the PlaySOM application starts playing a new song, either by a local user interaction or through

[3] `https://olymp.ifs.tuwien.ac.at/trac/PocketSOM/wiki/PocketSOMConnector`

a remote control client. *ePocketSOM* uses this information to display the current song on the screen and configure the control elements of the remote control accordingly. If playback is stopped, the song-id is empty.

- `playlist changed`
  When the playlist changes, the client is informed by this message. It can then retrieve the playlist from the server by requesting status information.

- `visualisation changed`
  When the visualization in the server application is changed, the information is propagated to the registered clients. The remote clients can load the visualization from the URL specified in the PSOM file.

#### Remote Control

The PocketSOM Connector provides a basic remote control interface to the internal audio player of *PlaySOM*. The client can submit commands to skip to the next or previous song, or start and stop playback. When starting playback, the client can either specify a song to play, or restart the song that was previously played.

Changes to the internal audio player are propagated to all connected clients, through notifications sent by the PocketSOM Connector, which can then update their internal state accordingly.

### 4.1.2 Playlist Handlers

A range of different playlist handlers are available by default. Moreover, the architecture is designed to easily allow to add further playlist handlers to support additional scenarios. All handlers share the basic functionality to process a playlist for a specific application. Some additionally provide advances interaction possibilities with the above described PocketSOM Connector.

**The M3U Saver** This playlist handler saves the created Playlist in a m3u file on local storage. The playlists can then be opened with a locally installed audio player or transferred to another device.

**The Local Player** The local player handler launches an external audio player installed locally on the device and transfers the playlist to it. While this handler sounds rather simple, implementing it within the restrictions of the mobile platform was quite challenging: The first versions of *ePocketSOM* were based on the Ewe VM, were

launching external applications did not reliably work. Therefore, a workaround was necessary, which works as follows: The playlist handler writes a temporary m3u file and a control file to the storage. This control file contains a sequence number and a command, either `stop` or `play` together with the path to the temporary playlist. A native standalone application implementing a simple watchdog timer, which periodically reads a command file for changes of the sequence number, then stopped or started the audio player accordingly.

After migrating *ePocketSOM* to the Eve VM, launching external applications works reliable, so the Local Player handler was rewritten to start the external audio player directly. This also improved the portability of the PocketSOM application, which was previously limited due to the dependency of the native watchdog timer.

**The VLC Client** The VLC handler was the first remote playlist handler. It uses the HTTP-interface of VLC media player[4] to send a playlist to a remote machine, i.e. the server. The songs are added to the VLC playlist, and then playback is started at the server, allowing PocketSOM to act as remote control. The VLC client only supports one-way communication – if any command sent to the player is lost, *ePocketSOM* does not notice the failure.

**The PlaySOM Connector** The most sophisticated playlist handler, the PlaySOM Connector, not only implements the features of a remote playlist handler like sending the songs of a playlist to a remote player and starting playback, but also provides advanced interaction features like sending a path drawn on the map on the mobile device to the remote PlaySOM application where the path is displayed on the map too, and full remote control over the internal player of PlaySOM. An overview of the architecture of the PlaySOM Connector is depicted in Figure 4.4.

When the PlaySOM Connector is enabled, a toolbar is added to the *LandscapePane* and the *PlaylistPane* providing advanced interaction control elements. These controls allow the user to start and stop playing, and switch to the next or previous song. All commands are executed in the PlaySOM application. Additionally the song currently played is shown. The connection between the *ePocketSOM* and PlaySOM is bidirectional, thus the mobile client is notified about events in PlaySOM like starting or stopping replay or skipping a song. Also changes to the playlist or the visualization can be propagated to the client.

Possible application scenarios enabled by these additional features will be further discussed in Section 4.2.

[4] `http://www.videolan.org/vlc/`

## 4.2 Applications and Scenarios

In this section, different applications and scenarios for PocketSOM will be outlined. Most of them are already implemented and available for the user, except the *Community Voted Playlist Generation* where the technical foundations are available, but the form of integration into the user interaction workflow is not yet decided upon.

### 4.2.1 Map-based Music Player

Most playlists on mobile devices are created and used to directly listen to music. The most important application therefore is to play music on the portable device. PocketSOM simply feeds its playlist to a local audio player in order to achieve this task.

Depending on the *Music Map*-settings, the local player can take the audio files from the device's storage or, if available, stream the music from a remote server using any available connection. In most cases a wireless-LAN connection will be used, but that depends heavily on the devices (and the player's) limitation. While local storage is limited, streaming allows access to a virtually unlimited number of songs, and PocketSOM can be directly connected to a large online audio portal.

### 4.2.2 Remote Control

A PocketSOM application can act as a remote control for selected audio players. Remote control for the VLC player is provided by the *VLC Client Handler*. The playlist is sent to the VLC player and added to the internal playlist. After that the VLC player starts playing. The VLC Client Handler was the first proof of concept for a remote playlist handler.

Based on the PocketSOM Connector, PocketSOM clients can gain full control over the internal player of *PlaySOM* after the user of the mobile device has been authenticated to the server. After that, paths drawn to the *Music Map* on the mobile device are sent to the PlaySOM application. Also the playlist can be synchronized to the PlaySOM application, and it is possible to start and stop playback, and skip to the next or previous song. Furthermore, the currently played song is shown at the client, and also changes to the playlist and the map (e.g. changing the visualization) are propagated to the mobile device.

### 4.2.3 Online Music Portal

The advantages of *Music Maps* in accessing large audio collections is not only applicable for private collections. The larger the collection, the higher the benefit of *Music Maps*, e.g. in an online music portal offering music as download or providing audio streams from a collection of some ten thousand items. Especially on mobile devices, where interaction is far more limited, this navigational approach is even more promising. The metaphor of a *Music Map* allows users (in this case customers) to explore the music offered in an interactive and intuitive way.

Songs are arranged similar to a common (offline) music store, where the music is organized by some form of semantic similarity (e.g. style, genre, geographical region, ...). In a record store, music from a certain style or genre are located together in on shelf, and the next shelf nearby contains music from a different, yet related genre whereas in the opposite corner of the shop a complete different musical style is presented. The shelves are labeled, and members of the staff can help with orientation and recommendation.

In classical online music stores, all the information about a song, its references to other songs that are implicitly available in a real world record store must be expressed and displayed explicitly by text, icons or cross-links. *Music Maps* display these additional information implicitly by the location of a song on the map of an online portal. Customers can focus and zoom into a specific region of the map, where they know their favorite music style is located and interactively explore the offered tracks. Depending on the business model, customers may be offered to download songs they select to their audio device, or, based on a subscription model, directly create an online audio stream with their favorite music by selection a region or drawing a path on the *Music Map*. The latter example is comparable to a personal radio station, playing music from ones favorite genres.

Creating a personal audio stream is also an alternative solution to the common problem of finding the right radio station. There exists a huge number of radio stations, both online and over the air, making the choice of an appropriate program difficult. The common way to find a radio station to one's liking, is to switch through several of them, listening to each for a couple of minutes until a convenient program is found. In [LR06], Lidy and Rauber propose a solution to this dilemma, by visually displaying the profile (or fingerprint) of a radio station on a *Music Map*, highlighting the regions on the *Music Map* where this station is active (i.e. where the music this station played recently is located). With this "fingerprint" one can easily identify a radio station to one's liking.

The method described in this thesis allows to create a radio station trough a visual profile instead of selecting an existing station based on a visual profile.

### 4.2.4 Autorouting

Instead of creating a playlist by drawing a path or selecting a region on the map, the user can select interesting points, or select a number of favorite songs ("landmarks"). PlaySOM will try to create a playlist of the desired length by connecting the points and songs to a path. As described in Section 3.1, the system might change the ordering of the submitted landmarks and insert additional songs between landmarks with larger a distance to provide a convenient playlist

An explorative utilization of this method could be to select two songs from different genres and see how the system conquers the distance. A playful example for the system could be the challenge to *"Get me from* Mozart *to* Shakira *within ten songs."* and then see (or better hear) which path the system chooses through the musical history and present.

### 4.2.5 Community Voted Playlist Generation

In most cases an online radio stream is created by a single user, sometimes by a small team. The audience usually has very limited possibilities to influence the content of the audio stream. Some radio stations offer the possibility to call the DJ and ask for a special song or genre.

By collecting paths drawn to different instances of a *Music Map* distributed over several devices, a radio stream can be created based on the audience's input. Every listener creates a path on the map which is then sent to a central server. There the paths are collected and then merged into one joint playlist using the methods presented in Chapter 3. The collective playlist is then made available to the community as online audio stream or directly played on an attached HiFi device for the general audience.

While the audio stream is played, the audience can further influence the forthcoming tracks. The position of the currently played song is show on the map, and so are the next 5 songs queued for playback. The audience can now nominate an alternative queue by selection songs or regions beside the current path, or by submitting new path segments. The system will then incorporate the submitted songs into the playlist. The audience can see the current nominations and the activities by the users interacting with the system and are thus invited to join the community by adding their nominations for the playlist. A quite similar service was demonstrated in 2007 by Leitich and Toth [LT07], however, without visual enhancements of *Music Maps* and paths.

## 4.3 Summary

*ePocketSOM* provides intuitive and interactive access to large audio collections on mobile devices. Playlists can be created by simply drawing a path on the map. A number of different modules is available to further process a created playlist by sending it to a local audio player, to a remote application or to the PocketSOM Connector of the PlaySOM application.

The PocketSOM Connector adds server functionality to PlaySOM. It provides the data files required by PocketSOM to display a *Music Map*, so mobile clients can directly connect and load a map from a central source, allowing multiple clients to access the same *Music Map* parallel. In combination with the PlaySOM Connector, *ePocketSOM* allows advanced remote interaction with the map by sending paths and playlists to the server and acting as a remote control for the server's internal audio player. The PocketSOM Connector also provides current status information such as the playlist, the song currently played or the selected visualization. Changes to the status information, such as a new playlist, a change of the visualization or the next song being played, are distributed to all connected clients.

This provides the prerequisites for new forms of interaction with audio collections and music consumption.

# 5 Playlists as Paths or Trajectories in Music Spaces

In this chapter the correlation between the quality of playlists and their visual shape on a *Music Map* is investigated. The quality of professional playlists (from a radio station), map-based playlists (c.f. Section 2.5.1) and playlists created by non-professional users was rated by the participants of a small scale user study and compared to the visual shape they create on a *Music Map*.

As source for user generated playlists last.fm, an online radio station was chosen. Last.fm allows registered users to listen to different, individually created radio streams by selecting a song, an artist or a genre. The system will create a radio stream starting with the selected song or artist followed by other artists the system perceives suitable. During each song, the user can rate a song as "loved" indicating he or she likes the song in the context of the current playlist, or "skip" the song to express a disapproval. The selection of adequate songs and artists is based on collaborative filtering mechanisms (c.f. Section 2.2.4) and the users listening (and love/skip) history.

Users can add songs to one of their personal playlists which are part of the account. Further it is possible to attach semantic tags (c.f. Section 2.2.2) or detailed comments to songs, artists and albums, and of course search for songs tagged with a specific tag. Last.fm also comprises basic social network functionality like becoming "friends" with other users. Users can view the profile of each other and browse their recently played songs and their playlists. For subscribed users[1], it is further possible to directly listen to the full playlist create by an other user, if the playlist contains at least $45$ songs from $15$ different artists.

Based on all the collected data, last.fm provides recommendations of new songs to users, and also the radio streams are based on, more generalized, recommendations. A playful realization of the recommendation system is the possibility to calculate the "musical compatibility" between two users.

For the experiments in this chapter, user created playlists from last.fm and playlists created by professionals, a popular mainstream radio station, were collected to create the data corpus. The quality of selected playlists from the corpus and playlists gen-

[1] subscribed users have to pay a monthly fee (3 € at the time of writing).

erated by the *PlaySOM* application, was rated in a small user study and the results compared with the visual representation of the playlists.

## 5.1 Data Corpus

To analyze and evaluate the behavior and shape of playlists on *Music Maps*, a collection based on last.fm users and their playlists was built. An officially supported API and publicly available preview snippets of most songs where used for building the collection.

Table 5.1: Playlists from last.fm

| Group | Count | | Coverage | # Songs total | # Songs unique | # retrieved Songs total | # retrieved Songs unique |
|---|---|---|---|---|---|---|---|
| all | 960 | | 71% | 59544 | 40203 | 42710 | 31772 |
| 80+ | 484 | (50%) | 90% | 21962 | 16176 | 19852 | 14297 |
| 90+ | 283 | (29%) | 96% | 11139 | 9177 | 10662 | 8736 |
| 100 | 82 | (9%) | 100% | 2690 | 2579 | 2690 | 2579 |
| 100-cut | 82 | (9%) | 100% | 897 | 884 | 897 | 884 |

The data corpus was created by randomly selecting approximately $900$ last.fm users who had created at least one playlist with more than $20$ songs. The playlists were fetched and saved, together with $30$ second snippets of the songs which are available for most tracks from last.fm. Songs that could not be retrieved from last.fm were fetched from amazon.com[2], where $30$ second preview snippets are also available.

This resulted in a data corpus of $31772$ audio files and $960$ playlists. A *Music Map* with $80 \times 60$ units was created for the visualization and analysis of the playlists. For all experiments in this paper, only playlists with a coverage of $100\%$ were used, reducing the size of the corpus of playlists from $960$ to $82$ playlists containing $2579$ unique songs. Coverage in this context describes the ratio of songs in the playlist that could be actually retrieved. The *Music Map* was, however, created with the complete corpus.

The corpus was further extended by adding playlists created by professionals. For $103$ hours, the song titles broadcasted by Ö3, a popular mainstream pop radio station[3], were logged, and again $30$ second snippets were fetched from last.fm and amazon.com. The playlists retrieved from the radio station were split at the full hour to

[2] `http://www.amazon.com`
[3] `http://oe3.orf.at`

Table 5.2: Playlists from the radio station (Ö3)

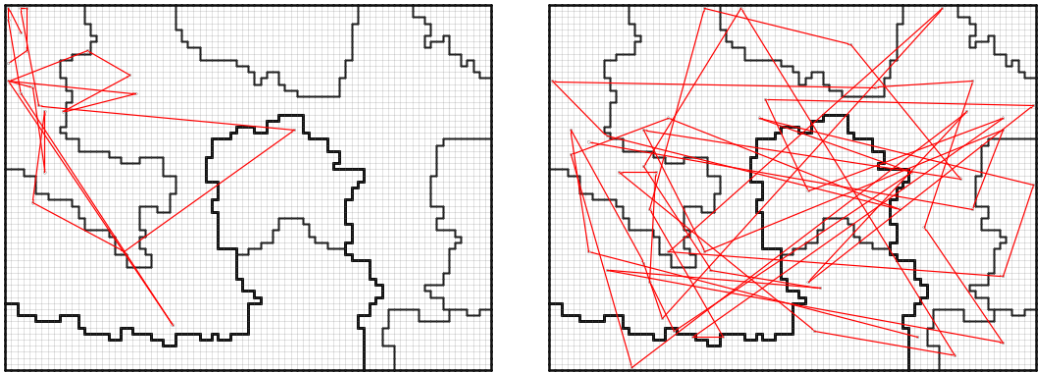
| Group | Count | | Coverage | # Songs total | # Songs unique | # retrieved Songs total | # retrieved Songs unique |
|---|---|---|---|---|---|---|---|
| all | 103 | | 92% | 1179 | 461 | 1094 | 406 |
| 80+ | 97 | (94%) | 94% | 1103 | 415 | 1043 | 379 |
| 90+ | 81 | (79%) | 96% | 923 | 380 | 891 | 356 |
| 100 | 49 | (48%) | 100% | 542 | 288 | 542 | 288 |
| 100-sel | 9 | (9%) | 100% | 117 | 103 | 117 | 103 |

avoid the gap created by news broadcasts. Within the $103$ logged hours, the radio station played $1179$ songs ($461$ unique songs), of which $406$ unique songs could be retrieved, covering about $92\%$ of the $103$ hours.

It is interesting to see that the radio-broadcasted playlists show a significant higher coverage of songs that could be actually retrieved from the data sources. The playlists from last.fm only achieved an average coverage of $71\%$, the radio-playlists reached $92\%$. On the other hand, for only $9\%$ of the playlists from last.fm all songs could be retrieved while for the radio-playlists it was possible for $48\%$. The fact, that most of the songs were fetched from last.fm – the "provider" of the playlists – even extends the disparity.

A possible explanation for this phenomenon is that the last.fm community has a more heterogeneous musical taste and includes more non mainstream music, whereas Ö3 has a more focused repertoire of music it is playing. This is also supported by the relative low number of unique tracks in the radio-playlists: The $1179$ tracks played in total were selected from a pool of only $461$ unique songs, in other words every song was played more than $2.5$ times on average. The five most frequently played songs occur $92$ times within the $1179$ tracks which is about every $13^{th}$ song. The $59544$ tracks from last.fm were selected from a pool of $40204$ unique songs, an average of $1.5$ times played. A song from the five most frequently played songs occurs about every $322^{nd}$ song.

To normalize the playlists from different sources, all playlists from last.fm of which all songs could be retrieved (Group "100" in Table 5.1) were truncated to the length of 13 Songs, creating the group "100-cut". From the radio-playlists with a with a coverage of 100% (Group "100" in Table 5.2) those containing exactly 13 Songs were selected to form the group "100-sel". From these two playlist pools the playlists for the user-study in Section 5.3 were selected.



(a) Homogeneous and focused playlist with some outliers

(b) Playlist without clear focus distributed over the entire map

Figure 5.1: Two playlists, visualized on a *Music Map*

## 5.2 Visualizing Playlists on *Music Maps*

To visualize a playlist on the *Music Map*, for each song on the given playlist the position on the map is located and linked with the positions of the adjacent songs in the playlist. This creates a path based on a playlist. Such visualizations can be used as a template for new playlists (showing them as example or recommendation to the user), but they also reveal specific and descriptive information about the playlist. In conjunction with different visualizations of the *Music Map* the shape of the playlist and the regions it covers allows conclusions about the musical style and variance of the playlist.

The map depicted in Figure 5.1 is a *Music Map* base on Rhythm Patterns with $80 \times 60$ units containing the complete corpus of $31772$ songs of the user study. The lines dividing the map into eight regions are cluster borders separating different musical styles from each other. For example, the cluster in the upper right corner mainly contains Hip-Hop and Rap songs. The cluster in the upper left corner contains mystical music from artists like *Nightwish* and *Vanessa Mae*. The cluster in the center of the map contains mainly rock/pop artists like *Ace of Base* or *The Smashing Pumpkins* but also several Hip-Hop and Rap songs, while the cluster below contains music from older artists like *Jimi Hendrix*. The emphasized border divides the map into the two main clusters with the largest differences.

The visual analysis of the playlists in the corpus showed very heterogeneous results. Some playlists clearly focused on a specific region on the map. The playlist shown in Figure 5.1(a) mainly comprises songs from artists like *Cult of Luna* or *Muse*. It shows

a clear focused shape and and stays in an area of the map. The outliers are caused by songs like "How to Disappear Completely" by *Radiohead* or "Your Hand in Mine" by *Explosions in the Sky*, separated by larger steps from the main part. Figure 5.1(b) provides as an excellent counter-example. The playlist, containing a colorful mixture of various artists like *Nirvana*, *The Doors* or *Elliot Smith*, covers almost the entire map, jumping from one corner to the complete opposite. It further shows no clear focus for any region on the map.

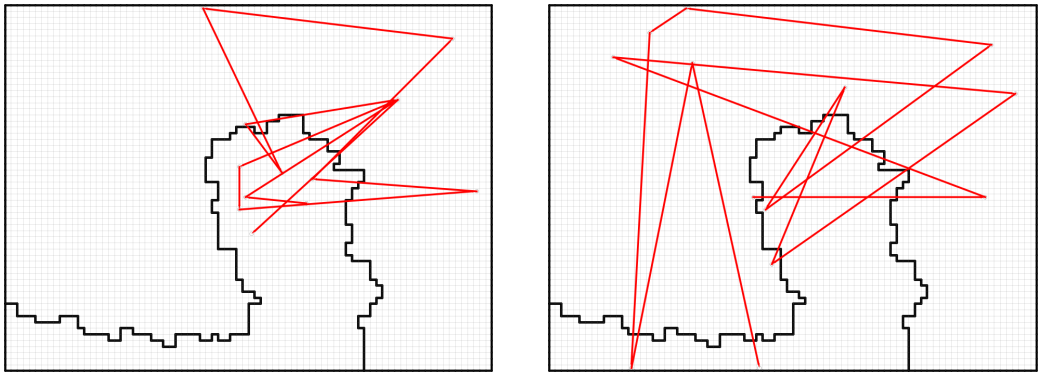
Unfocused and distributed playlists are the clear majority in the corpus. Only a handful of playlist formed a continuous path or focused onto certain areas on the map. Further investigations of the playlists and their paths showed, that many playlists from last.fm that focused on specific regions only contained tracks from one artist, sometimes even only from one album, which is uncommon for most playlists, thus raising the questions whether the playlists retrieved from last.fm can be considered as "real" playlists, or if they are used as some kind of "bookmark list" for favorite songs. But also many of the radio playlists did not show a clear focus on the map.

## 5.3 User Study

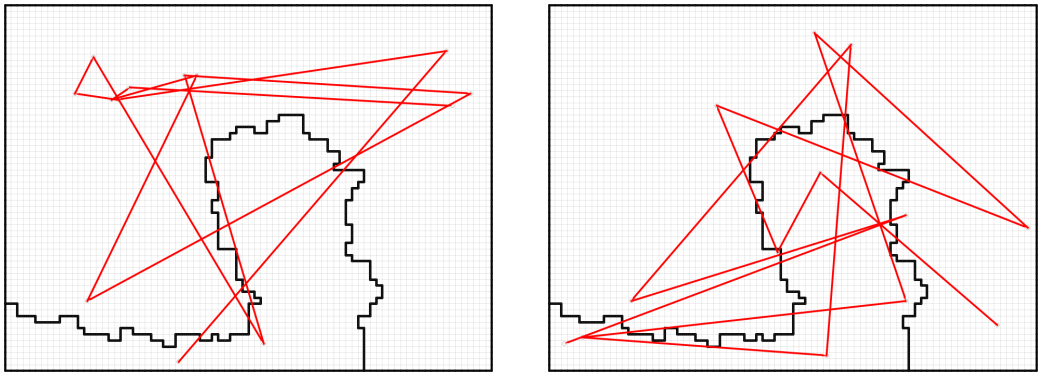
To further investigate the quality of playlists from different sources (user-created playlists from last.fm, professional playlists from a radio station and map-generated playlists), and to determine whether the visual shape of a playlist allows conclusions about the quality of a playlist, a small-scale user study was launched. Two playlists were chosen from the pool of available last.fm-playlists based on their graphical representation on the map, one with a focused visual shape and one with covering a large part of the map (c.f. Figure 5.1). Further also two playlists from the radio station were picked analogous to the playlists from last.fm. Finally, two playlists generated through the *Music Map* (one by drawing directly a path onto the map and one by applying the routing algorithm from Section 3.1 on several selected units) were included in the study. The visual shapes of the selected playlists are depicted in Figure 5.2.

Depending on the map, and to which properties the feature sets used are sensible to, the visual shape of a playlist differs. Figure 5.3 shows two playlists from the user study on three maps created with different feature sets (RP, SSD, RH+SSD. While the individual visual shape is different on each map, the general shape (distributed or focused) is the same on all maps. The map to the very left was used for the investigations in the user study, but since the general shape is about the same on all maps the results should be representative for all feature sets.

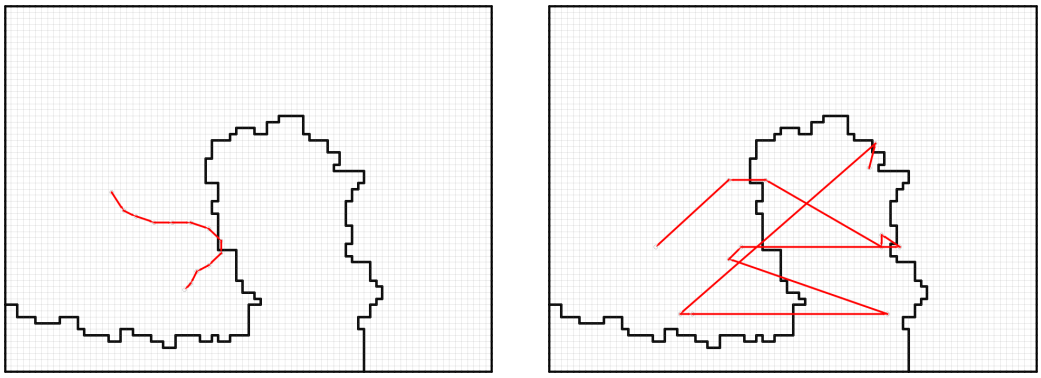
Before given to the participants, all the selected playlists were randomly ordered and renamed to conceal their source. For each of the six different playlists, the partic-



(a) last.fm (focused) (b) last.fm (distributed)

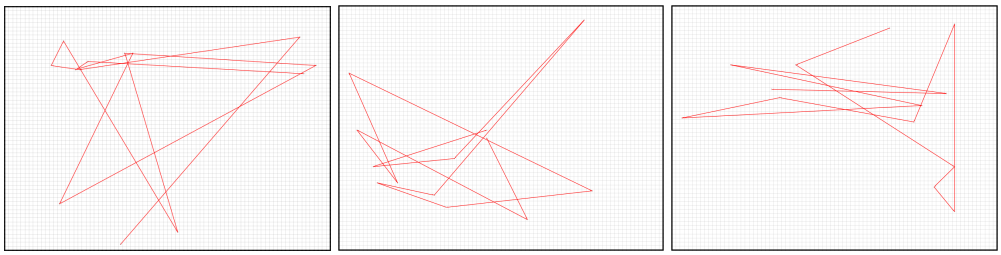


(c) radio (focused) (d) radio (distributed)

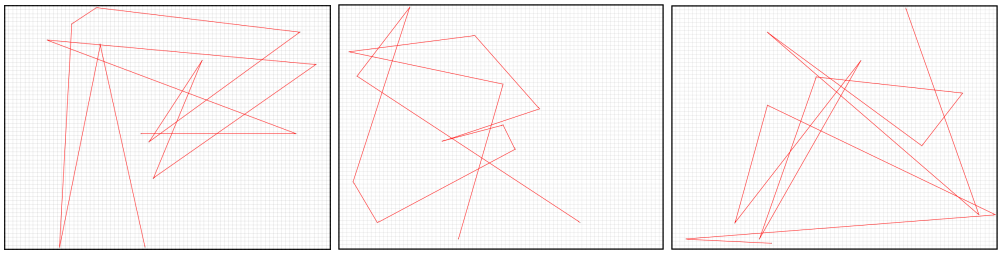


(e) generated (focused - path) (f) generated (distributed - routed)

Figure 5.2: Visual shapes of the playlists used in the user-study



(a) Playlist radio (f)



(b) Playlist last.fm (d)

Figure 5.3: Comparison of the visual shapes of two playlists on three maps based on different feature sets (RP, SSD, RH+SSD)

ipants were asked to listen to the music and fill out a questionnaire (see Appendix A.2) asking to

- rate how good a song fits to its preceding song, on a scale from $1$ to $5$, where $5$ represents the best value,
- give an overall rating of the playlist on the same scale,
- select up to three songs from the playlist that should be removed to improve the quality, and
- name situations and/or locations where this playlist would fit in.

The results are presented in Section 5.3.2.

### 5.3.1 Participants

The questionnaire was completed by five participants (three male, two female). A user study in this size must not be considered as a representative evaluation, it is more an indication of potential issues with the playlists.

Table 5.3: Average transition ratings by the participants. $t$ is the average transition rating, $o$ is the overall rating of the playlist. Playlists marked with (f) show a focused shape on the map, (d) denotes distributed playlists. Visual shapes are depicted in Figure 5.2

| Playlist | P-1 (m) | | P-2 (m) | | P-3 (f) | | P-4 (m) | | P-5 (f) | | average | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $t$ | $o$ | $t$ | $o$ | $t$ | $o$ | $t$ | $o$ | $t$ | $o$ | $t$ | $o$ |
| last.fm (d) | 2.25 | 2 | 1.67 | 1 | 2.00 | 3 | 2.17 | 2 | 2.67 | 2 | 2.15 | *2.0* |
| last.fm (f) | 2.75 | 3 | 2.17 | 3 | 2.92 | 1 | 3.00 | 3 | 3.67 | 4 | 2.90 | 2.8 |
| radio (d) | 2.50 | 4 | 3.17 | 4 | 2.50 | 4 | 1.67 | 1 | 2.92 | 3 | 2.55 | 3.2 |
| radio (f) | 3.42 | 4 | 3.33 | 4 | 3.00 | 3 | 2.83 | 2 | 4.33 | 5 | **3.38** | **3.6** |
| generated (d) | 2.17 | 2 | 1.67 | 2 | 2.08 | 3 | 1.67 | 2 | 2.75 | 2 | *2.07* | 2.2 |
| generated (f) | 2.67 | 2 | 2.42 | 3 | 3.50 | 4 | 2.33 | 3 | 3.33 | 4 | 2.85 | 3.2 |

Table 5.4: Misfit ratings per playlist. The table is to read as follows: In playlist "last.fm (d)", $14$ misfit votes were placed to $8$ different songs, where the top three misfits received $64\%$ of the votes

| Playlist | misfit votes | unique songs | top 3 |
|---|---|---|---|
| last.fm (d) | 14 | 8 | 64% |
| last.fm (f) | 9 | 6 | 66% |
| radio (d) | 9 | 7 | 56% |
| radio (f) | 13 | 9 | 46% |
| generated (d) | 12 | 9 | 50% |
| generated (f) | 15 | 6 | 80% |
| Total | 72 | 45 | 18% |

All participants are between 20 and 30 years old and German speaking. All of them have attended natural science studies at a university; three of them already received a master's degree, one is still a student and one quitted the studies without degree. Three of the participants are currently affiliated with a university or research institution, one is a student and one is working as software developer. Three participants are living in Austria (two male, one female), two in Germany (one male, one female).

### 5.3.2 Results

Here, a short summary of the key findings of the user study is provided, followed by a more detailed discussion in Section 5.4. The responses to the questionnaire, summarized in Table 5.3 and 5.4, lead to several interesting conclusions:

1. Individual participants gave coherent answers. In most cases, the average rating of the transitions was about the same as the overall rating of the playlist.
2. The participants showed a clear preference for "professional" playlists taken from a radio station over all other playlists (c.f. Table 5.3).
3. For the overall rating, playlists generated by the PlaySOM application performed better than the playlists retrieved from last.fm, while for the individual transitions, the playlists from last.fm were slightly in favor.
4. Comparing playlists from the same source to each other, the one with the more focused visual shape (marked with **(f)**) in all cases got higher ratings for the individual transitions and the overall rating.
5. Whereas the individual rating of the transitions differ in most cases, the participants clearly agreed when naming the songs that would not fit into the playlist and thus should be removed: $60\%$ of the misfit votes were given to $20\%$ of the songs, further $6$ songs were rated as misfit by more than $50\%$ of the participants. (c.f. Table 5.4).

## 5.4 Interpretation

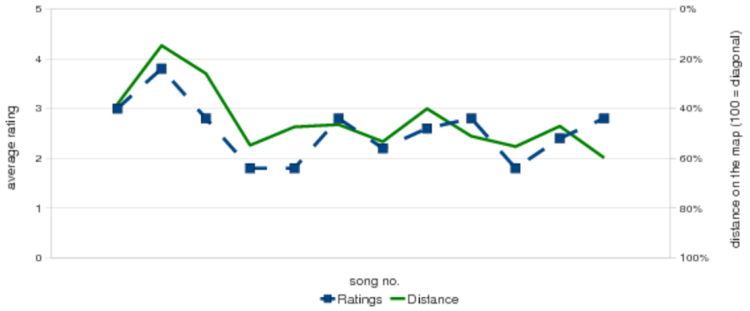
The following section provides a closer look into the playlists and their genesis, followed by an interpretation of the results from the user-study.

The figures in this section show charts of the transition ratings, distances and misfit ratings for each playlist. For each song the results of the questionnaire is depicted according to its position in the playlist. The blue dotted line represents the average transition ratings by the participants (referencing to the left Y-axis) while the solid green line shows the distance on the map (right Y-axis, $100\%$ is equivalent to the map's diagonal). In the second series of charts, the red diamonds (right Y-axis) represent the number of participants who would have removed this song from the playlist to improve the overall quality of the playlist.

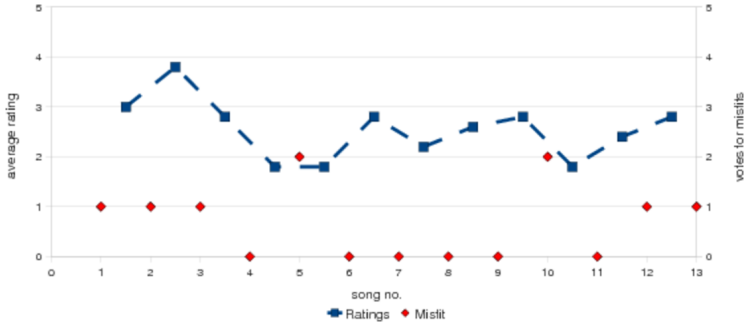
To further validate the results of this user study, all experiments should be repeated based on a larger data corpus containing more diverse musical styles. Due to the sources of the current corpus, pop and rock music is heavily dominating.

### 5.4.1 Radio (d)

Playlist *radio (d)* (see Appendix A.1.6) was recorded from Ö3 on May, $12^{th}$ 2009 from 10 to 11 am. The program at this time of the day is called "Ö3 Extra" and is designed




(a) the graph shows a quite high correlation between the distances and the ratings




(b) misfits in the playlists correspond with low ratings

Figure 5.4: Selected results from the questionnaire for *radio (d)*

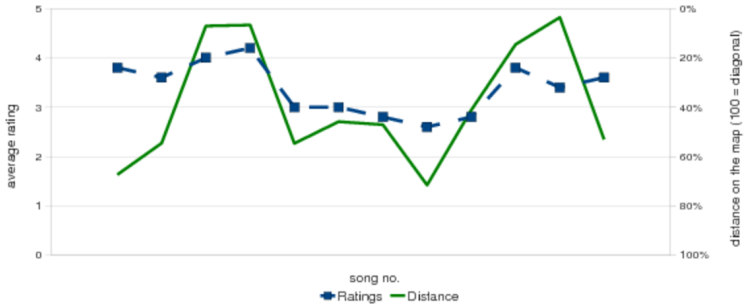
to accompany people at work, in an office as well as in a laboratory or workshop. Most of the time is dedicated to music interrupted by short news stories from different regions of the country. At the half hour is a short news headline announcement and at the full hour a five minutes news broadcast, both preceded by commercials.

In this playlist, the ratings and distances on the map express a similar trend, which is observable in Figure5.4(a). Further not a single song received enough misfit votes to be counted as an outlier (see Figure 5.4(b)). However, we will take a closer look at those two songs with the highest misfit score: The fifth song, “Lasse redn” by *die ärzte*, was apparently recorded at a higher overall volume level compared to the other songs in the playlist and contains German lyrics, both possible reasons to be qualified as outlier. The tenth song, “Sacrifice” by *Elton John*, does not fit into the playlist since it is a rather moving ballade compared to the other, more joyful songs. Such silent and slow songs are very often played after a sad and dramatic report, which was probably the case here.

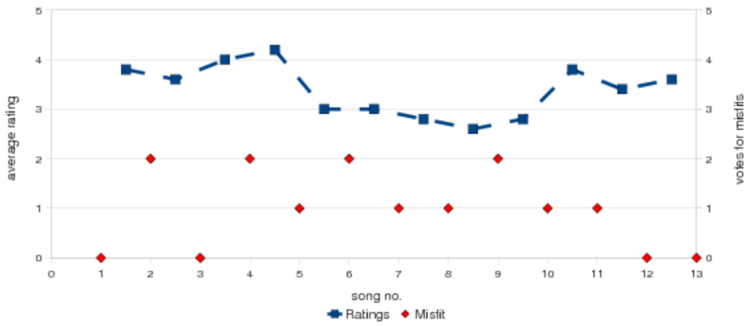
Around both songs, neither features nor map did express a significant variation of

the step-width, which is not surprising because (a) the features used are not sensible for different languages in the lyrics and (b) the ground-rhythm of the ballade was about the same as in the other songs.

### 5.4.2 Radio (f)




(a) Lower distances on the map are accompanied by higher ratings




(b) In this playlist no dominant misfits were detected

Figure 5.5: Selected results from the questionnaire for *radio (f)*

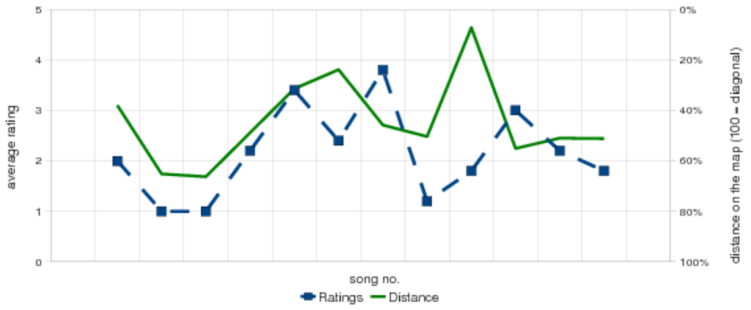
Playlist *radio (f)* (see Appendix A.1.3) was recorded from Ö3 on May, $7^{th}$ 2009 between 10 and 11 pm. The scheduled program for this period is called "Liebe usw..." ("love etc..."), a show about romance and love. The program consists primarily of music, and some listeners calling in and talking about their private and romantic relations. There are no commercials and only a three minute news broadcast to the full hour.

The playlist consists of 13 slow romantic songs, mainly ballades, from the last 30 years of mainstream pop music. All songs in this playlist received a misfit rating below three, thus no one is counted as outlier. The visual shape of this playlist on the map, depicted in Figure 5.2(c), shows a main group of seven songs in the upper left

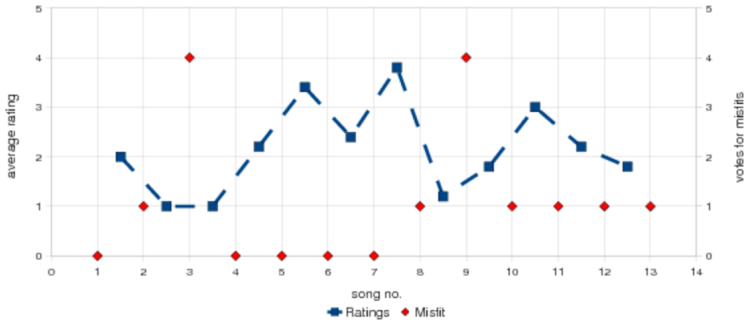
area and a minor group of three songs in the upper right. The three remaining songs are distributed in the lower part of the map.

Four songs in this playlist were rated twice as misfitting (see Figure 5.5(b)), one of them ("Run" by *Leona Lewis*) is located within the upper left main group, one ("You Sang to Me" by *Marc Anthony*) is in the upper right minor group (but still clearly separated), the remaining two, "'74-'75" by *Connells* and "Halo" by *Beyoncé*, are located at the bottom of the map. Both songs notably differ from the rest of the playlist. The song '74-'75 is apparently a karaoke version of the original song, Halo is a very rhythmical song whereas the other songs are rather slow ballades. It is further noteworthy that besides the one located in the main group, these songs have long distance connections to their adjacent songs. Figure 5.5(a) shows that the distances cover a broader range but still distances and ratings again share the same trend.

### 5.4.3 Last.fm (d)




(a) The songs 8 through 11 break the correlation between distance and rating




(b) Misfits (songs 3 + 9) are surrounded by low ratings

Figure 5.6: Selected results from the questionnaire for *last.fm (d)*

Playlist *last.fm (d)* (see Appendix A.1.1) was created by a last.fm user from the USA. Since October 2005, when she joined last.fm, she played more than $45.000$ songs which is about $30$ songs per day on average. Her top ranked artists are *Björk, Cut Copy, Nine Inch Nails* and *ADH*. The playlist she created on last.fm is titled "Something Awesome".

The playlist is a varied mixture of songs from very different styles, genres and epochs. The major outliers according to the misfit ratings are two songs, one ("Krzesany" by *Wojciech Kilar*) would fit as background music in a horror movie, the second, "St. James Infirmary" by *Cab Calloway*, could be from the 1930s. $80\%$ of the participants agreed that these two songs do not fit into this playlist. Visually, these songs are not separated on the map (see Figure 5.2(b)) because the playlist is distributed over the whole map due to the heterogeneous mixture in this playlist. Furthermore the ratings in the questionnaire range from quite good to very low values, but still both distances and ratings express the same trend (see Figure 5.6)

In conclusion, one could say that *last.fm (d)* seems to be not a playlist in a classical meaning but more a bookmark-list for a bunch of favorite songs.

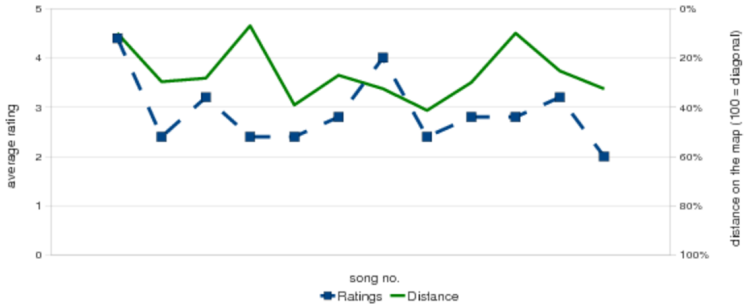
### 5.4.4 Last.fm (f)

Playlist *last.fm (f)* (see Appendix A.1.2) was created by a last.fm user from Finland, who uses last.fm on a regular base. Since October 2006, when she joined last.fm, she listened to more than $29.000$ songs, an average of about $25$ songs per day. Most of the music she's listening to is tagged as *indie, alternative, rock, electronic, Finnish, female vocalists* or *hip-hop*. Some of her most frequently played songs are performed by *Nujabes*, *The Go! Team* and *Dir en grey*.

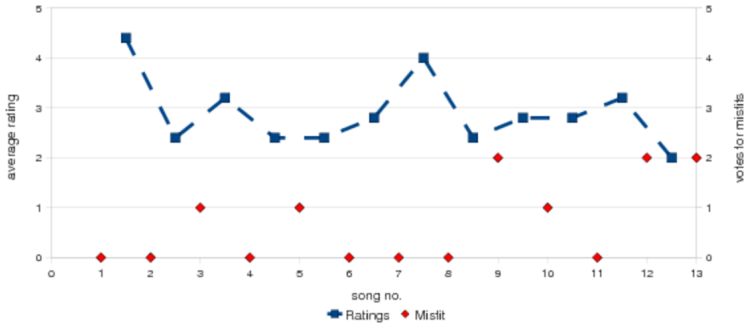
The playlist mainly contains hip-hop songs, but also some other styles, e.g. electronic music. One song appears twice in the playlists at position 3 and 9, listed under different titles[4] and thus was not recognized as duplicate. On the map, however, both files of the same song are located in the same unit.

The double appearance of the song confused some of the participants, others didn't notice. It is especially interesting that the second appearance in the playlist got higher misfit ratings than the first (see Figure 5.7(b), song no 3 + 9), leading to different possible explanations. One could be that not the song as such does not fit into the playlists, but the repetition itself. Another explanation could be that the first appearance simply fits better into the surrounding tracks. This is further supported by the fact, that the first appearance also received (slightly) better transition ratings (and lower distances)

[4] the song was listed as *Quantic - Time is the enemy* and *Time is the enemy - Quantic* and also with different filenames.




(a) *Last.fm (f)* shows a low correlation between distance and ratings




(b) The misfitting songs (9,12,13) correspond with the lower ratings in this playlist

Figure 5.7: Selected results from the questionnaire for *last.fm (f)*

to its neighbors.

The visual shape of the playlist (see Figure 5.2(b)) corresponds to the musical styles of the songs. Songs typical for hip-hop are located in the middle part of the map, while the other songs are separated. A reggae-song, "Could You Be Loved" by *Bob Marley*, is located at the top edge of the map. Also the other songs of a notable different style (e.g. "Didn't Cha Know" by *Erykah Badu*, a rather melodic song) are clearly separated from the main group and also placed in a different cluster while both snippets of the same song are located on the same unit. Also the transition ratings between songs from the main group and the "outliers" reflect the visual shape: Transitions from and to outliers did receive lower ratings than transitions within the main group (see Table 5.5), except for the transition from the main group to the song in the upper right corner ("Luchini" by *Camp Lo*) which received the highest rating.

Table 5.5: Average ratings between song clusters on the map for the playlist *last.fm (f)*

| group | songs | main group | outlier 1 |
|---|---|---|---|
| main group | (1,2,4,5,7,10,11,13) | 3.2 | 2.8 |
| outlier 1 | (9) | 2.8 | – |
| outlier 2 | (8) | 4.0 | 2.4 |
| outlier 3 | (3,12) | 2.7 | – |
| outlier 4 | (6) | 2.6 | – |
| outlier 1-4 | (3,6,8,9,12) | 2.9 | – |

### 5.4.5 Routed

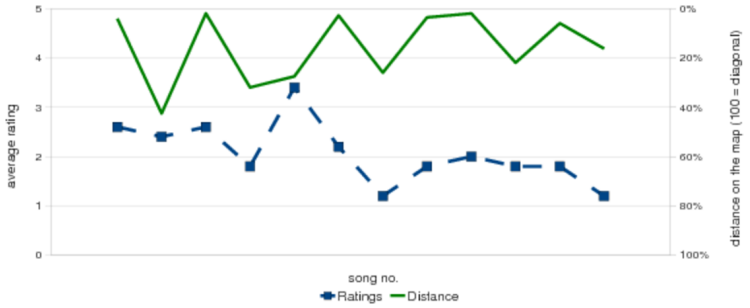
Playlist *routed* (generated (d), see Appendix A.1.4) was created by randomly selecting seven songs on the map and then applying the routing algorithm from Section 3.1 (based on the feature space) which inserted one song between each pair of successive songs of the previously selected songs. In the resulting playlist, songs at an odd position were selected manually, songs at even position were selected by the routing algorithm and are located in the middle between its neighbors. These calculations were based in the input space to create more convenient transitions.

The major outlier (see Figure 5.8(b)) in this playlist is the seventh song ("Metal Gods" by *Judas Priest*), a song from the heavy metal genre embedded into the a series of calmer songs, e.g. "Happiness Is a Warm Gun" by *The Beatles* or "Wide Open Spaces" by *Dixie Chicks*. The transition from The Beatles to Judas Priest is a steep change in style, but the song in between ("Washer" by *Slint*) selected by the algorithm does support the passage quite well.

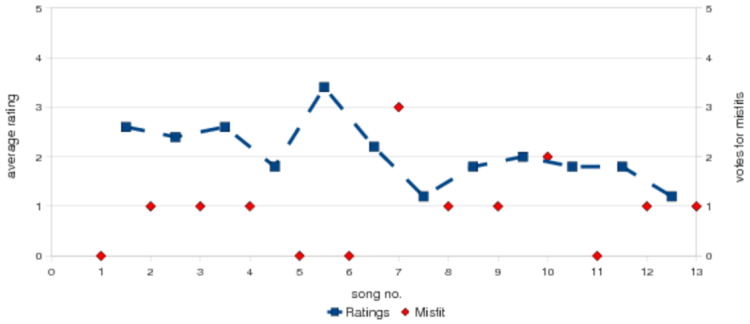
The visual shape of this playlist is depicted in Figure 5.2(f). The playlist covers a large area on the map and contains short steps as well as long distances between the individual songs. It is interesting that, based on the map layout, in most cases short and long distances alternate. Only twice, two long (or short) steps appear in a row. In this playlist, a correlation between the distance on the map and the rating of the transitions can be observed. Short steps did receive better ratings than long steps when compared locally. In the overall view of the playlist, the difference is not significant (see Figure 5.8(a)).

### 5.4.6 Line

Playlist *line* (generated (f), see Appendix A.1.5) was created by drawing a path on the map. 13 songs were randomly selected along the path, distributed over the full length of the line. This is also reflected in Figure 5.9(a) showing constantly low distances




(a) The routing algorithm produced alternating step-widths



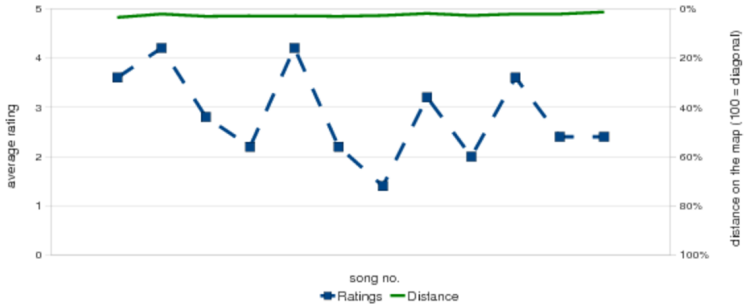

(b) The major outlier (song 7) received the lowest rating

Figure 5.8: Selected results from the questionnaire for *generated (d)* (routed)

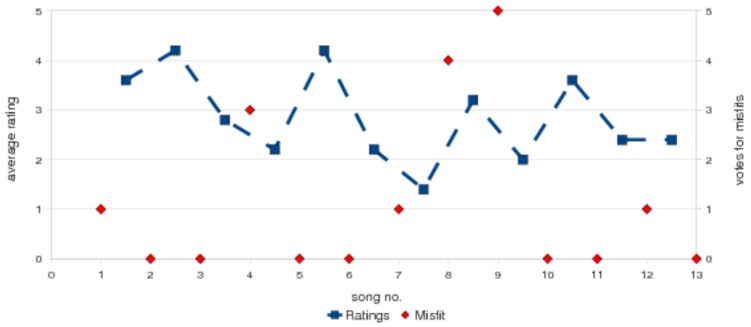
between the songs. Towards the end, the playlist closely moves towards the border between the two main clusters which is even crossed for one song (see Figure 5.2(e)). While the distance on the map between these tracks are within the normal range of this playlist, the musical style should change abrupt.

The first song rated as a misfit in this playlist (see Figure 5.9(b)) is "An Enemy Led the Tempest" by *Cradle of Filth*, which is of a complete different and outstanding style compared to the other songs in the playlist: *Cradle of Filth* is from the genre "dark metal" whereas the most other songs are from the genres "punk" and "rock". Regarding the rhythmical structure, however, the song does fit into the series of songs although the genres differ.

The second rated misfit is a pair of two songs in sequence that introduce a completely different musical style ("electronic") into the playlist, on a semantic as well as on a rhythmic level. The two songs, "Ceremony" by *Empusae* and "Allahi Allah" by *Niyaz*, have been rated to fit together quite well, but the transitions of both to their neighbors received the lowest ratings in the whole playlist. A detailed examination of the visual shape of this playlist in Figure 5.2(e) shows that these two songs are those




(a) A line-based playlist has constant step-widths




(b) The misfits clearly identify songs from different clusters (song 8,9)

Figure 5.9: Selected results from the questionnaire for *generated (f)* (line)

songs close to the cluster border.

## 5.5 Summary

To investigate the connection between the quality of playlists and their visual shape on *Music Maps*, a small user study was launched asking the participants to judge the quality of playlists from different sources.

The corpus of playlists was created by retrieving personal user-created playlists from last.fm, an online music portal. Further, playlists from a mainstream radio station were used as reference for playlists created by professionals. Playlists created with the *PlaySOM* application completed the corpus. The tracks contained in the playlists where fetched as 30 second preview samples from last.fm and amazon.com. It was interesting to see, that significantly more tracks from the radio-playlists could be retrieved than from the playlists from last.fm

The results of the user study suggest a correlation between two songs matching

together in a playlist and the step-distance on the map. Playlists with a more focused shape in all cases also received higher ratings, whereas playlists "jumping" on the map from corner to corner received lower ratings in the user study.

Many playlist with a distributed shape tendentially resembled more a collection of songs than a consciously created playlist, but also some playlists created by professionals contained large steps and covered significant distances on the map.

To sum up, the path metaphor of creating a playlist by drawing a path onto a *Music Map* in principal proved itself useful, but it is not necessarily the only way to create a good playlist. Of course, carefully created manual playlists receive better ratings, but on average, playlists that have been automatically generated through the map can compete with user created playlists.

# 6 Conclusions & Future Work

In this thesis different schemes for multi user interaction with music libraries have been presented, compared and discussed. Chapter 1 gave an insight into the motivation and challenge of novel interaction methods with digital music libraries. In Chapter 2, an overview on related work to this thesis has been given, including different audio feature sets, music organization methods and applications.

Chapter 3 presented a set of methods to combine and merge playlists from different users to enable a basic form of collaborative playlist creation. Based on the path metaphor of playlists on *Music Maps*, playlists are split, merged and concatenated into a joint playlist for the audience. The visual shape of the playlists on the *Music Map* will reflect the progress and result of the combination process enabling the listeners to see and understand the changes made to their playlist, also raising the acceptance of the new playlist.

In Chapter 4, ePocketSOM, an application to view and interact with a *Music Map* on mobile devices, was presented. Multiple instances of ePocketSOM can connect to a central server providing a shared *Music Map*. Playlists created on the mobile clients are shared with the server and further with the community connected to it. ePocketSOM can act as remote control device for the PlaySOM application, creating new playlists and controlling the internal audio player. Vice versa, the mobile clients can load the playlist from PlaySOM and connect to the shared playlist in a online audio stream like setting.

Chapter 5 investigated the relation between the quality of playlists and their visual shape on *Music Maps*. A data corpus of more than $31.000$ songs and more than $1.000$ playlists, created by users and professionals, was created. From each source, one playlist with a focused shape on *Music Maps*, and one distributed playlist together with two playlists generated by the *Music Map* were evaluated in a small scale user study. The results showed a clear preference for the playlists created by professionals, but the focused playlists were favored over the distributed. This suggests the assumption that there exists a connection between the quality of a playlist and it's visual shape on the *Music Map*.

## 6.1 Applications and Installations

The metaphor of playlists as paths on a map or in a landscape enables further playful interaction forms beyond the creation of playlists itself. The possibilities range from art installations to mobile social network interaction [FLP+09]. In the following, some emerging applications will be presented. While *Music Maps* provide intuitive, interactive and playful access to large music collections, displaying them on a computer screen again raises a barrier for users to understand and accept the concept. Images shown on a screen are abstract and thus the concept of creating a playlist through a paths also stays abstract.

### 6.1.1 *Music Maps* in the Real World

One way to bring *Music Maps* closer to the people and make the concept of Music Maps coming alive is to materialize a *Music Map* in the real world. The main idea is to transform a Music Map from the screen representation to a real world area, e.g. in a *MusicSOM Cafe* [FLP+09].

The basic concept of a real world *Music Map* consists of a public or private space with several loudspeakers distributed in it. Each speaker represents a position on the map and plays the songs associated with this position. The relative position of the speakers to each other corresponds to the position it represents on the map. The speakers playing simultaneously create an acoustic representation of a *Music Map* in the real world.

For the MusicSOM Cafe, the speakers are placed on, or integrated into tables. Visitors can now walk between the tables to explore the musical space and the range of different music styles offered in the room, and search the table where they want to drink their coffee. There they will meet people with similar music preference. Exploring other musical styles is easily possible by just changing the location and listening at another table.

A group formed by people gathering around a table of a certain musical style could start a discussion about the direction in which to move on to the next table, thus implementing *collaborative music consumption through mobile listeners*.

### 6.1.2 *Music Maps* in Virtual Environments

Instead of placing a *Music Map* into the real world, it is possible to virtually place the user on the *Music Map*. various techniques exist ranging from wide spread ones such as 3D-computer games to expensive and technically demanding approaches like a CAVE virtual environment [CNSD+92].

In a virtual environment, the user (or the avatar depicting the user) can walk through the landscape while songs from his current position is played constantly. This allows the user to explore the map and music collection. By including techniques from multi-player online games users can also explore the map in groups and exchange themselves about the music they hear.

### 6.1.3 Ambient *Music Maps* on Mobile Devices

With mobile devices, a *Music Map* can be virtually spread over a large area, such as a park or even an entire city. The shape of the map can be adopted to the shape of the target area [MMR05]. With their mobile devices, users can now listen to music that corresponds to their position on the map. By choosing different routes, e.g. for their way from home to the office, users also choose different incidental music. Depending on the music, some users might choose to take a small detour to avoid a certain musical style, or to raise the probability for a specific song. In this scenario, also the map could adjust itself to the behavior of the users. By matching frequent routes of a user with his favorite songs the system can adapt the map.

Another scenario based on *Music Maps* and mobile devices could be the following. A central map is provided where users can create playlists by selecting regions or paths and then listen to music while they are on their way. The mobile device can then point out other users whenever the virtual (musical) position and the real world position of two users get close. So the system will display other users near me, e.g. within 100 meters, that are currently listening to the same style of music. This can be even expanded to some kind of multi-dimensional tag-play.

## 6.2 Outlook and Future Work

So far, the system presented is in the status of a prototype. Improvements are possible in various parts of the system, some major tasks that will be addressed in the near future are described in the following lines.

**Basic Concepts** One task will be the improvement of the basic concepts used with *Music Maps*. This might involve new feature sets expressing properties of the audio that have so far been ignored such as time-dependent features, or integrating multi-modal features – not only audio, but also lyrics, album covers, music videos, artist biography or user comments.

On the other hand, also the process of map creation has potential for improvement. So far, only little effort has been put into the selection of the distance function used

in the training process. In most cases the euclidean metric is used, but e.g. a feature specific distance function that takes the properties of the features into account might lead to significantly improved results.

**Propagation** A major problem for all investigations based on *Music Maps* is that the installation of the framework and the creation of a map is technically challenging. Because of this, the user community is too small to receive feedback or participants for a large scale user-study. To propagate the concept of *Music Maps*, on the one hand it is important to provide an easy installable and usable version of software. On the other hand it is necessary to bring PocketSOM to mobile devices of many different platforms such as *Android*, *iPhone 3.0*, *Maemo* or *Symbian*, and provide system specific additions for each of them: the new version of the iPhone OS provides access to the internal music library which could be used for a local *Music Map*. A third way to propagate *Music Maps* could be by a cooperation with a content provider, such as an online music store or a online radio station.

**Multi-User Features** Also the multi-user features and scenarios will be part of further investigations. This includes technical limitations such as the current role of a central server which might be removed for ad-hoc connections between multiple devices, but also techniques to merge two or more different *Music Maps* in case some users spontaneously want to share their music collection in a jam-session style.

# Final Remarks

*Music Maps*, providing an alternative access to large music collection, have the potential to fundamentally change the way people interact with music. Listening to music can change from an individual act to a process embedded into a social network. Playlists are not created by a single person but collaboratively created by a group sharing common interests, in discussion and negotiation. Online radio streams adapt themselves to the preferences of their audience automatically. To develop systems and applications that can support this novel access towards music will be a challenge in the upcoming years.

# Bibliography


[AAG06] Xavier Amatriain, Pau Arumi, and David Garcia. CLAM: a framework for efficient and rapid development of cross-platform audio applications. In *Proceedings of the 14th annual ACM international conference on Multimedia (MULTIMEDIA '06)*, pages 951–954, New York, NY, USA, 2006. ACM.

[ABA05] Mohammed Attik, Laurent Bougrain, and Frédéric Alexandre. Artificial Neural Networks: Biological Inspirations. In *International Conference on Artificial Neural Networks (ICANN '05)*, volume 3696/2005 of *Lecture Notes in Computer Science*, pages 357–362. Springer Berlin / Heidelberg, 2005.

[AT01] Masoud Alghoniemy and Ahmend H. Tewfik. A network flow model for playlist generation. In *International Conference on Multimedia and Expo (ICME 2001)*, pages 329–332. IEEE Computer Society, Aug. 2001.

[CNSD+92] Carolina Cruz-Neira, Daniel J. Sandin, Thomas A. DeFanti, Robert V. Kenyon, and John C. Hart. The CAVE: audio visual experience automatic virtual environment. *Communications of the ACM*, 35(6):64–72, 1992.

[DEH05] J. Stephen Downie, Andreas F. Ehmann, and Xiao Hu. Music-to-knowledge (M2K): a prototyping and evaluation environment for music digital library research. In *Proceedings of the Joint Conference on Digital Libraries (JCDL '05)*, page 376, Denver, Colorado, USA, Jun. 2005.

[Dow03] J. Stephen Downie. *Annual Review of Information Science and Technology*, volume 37, chapter Music Information Retrieval, pages 295–340. Information Today, Medford, NJ, USA, 2003.

[Dow08] J. Stephen Downie. The music information retrieval evaluation exchange (2005–2007): A window into music information retrieval research. *Acoustical Science and Technology*, 29(4):247–255, 2008.

[FLHR08] Jakob Frank, Thomas Lidy, Peter Hlavac, and Andreas Rauber. Map-based music interfaces for mobile devices. In *MM '08: Proceeding of the 16th ACM international conference on Multimedia*, pages 981–982, New York, NY, USA, Oct. 2008. ACM.

[FLP+09] Jakob Frank, Thomas Lidy, Ewald Peiszer, Ronald Genswaider, and Andreas Rauber. Ambient Music Experience in Real and Virtual Worlds Using Audio Similarity. *Multimedia Tools and Applications*, 44(3):449–468, Sep. 2009.

[Fra08] Jakob Frank. Enhancing music maps. In *Proceedings of the 8th International Student Workshop on Data Analysis (WDA 08)*, pages 53–59, Jun. 2008.

[Fra09] Jakob Frank. Analysing and evaluating playlists on music maps. In *Proceedings of the 9th International Student Workshop on Data Analysis (WDA 09)*, pages 10–17, Jul. 2009.

[FSGW08] Arthur Flexer, Dominik Schnitzer, Martin Gasser, and Gerhard Widmer. Playlist generation using start and end songs. In *Proceedings of the 9th International Conference on Music Information Retrieval (ISMIR '08)*, pages 173–178, Philadelphia, USA, Sep. 14-18 2008.

[GCM08] Derek Green, Pádraig Cunningham, and Rudolf Mayer. *Machine Learning Techniques for Multimedia*, chapter Unsupervised Learning and Clustering, pages 51–90. Cognitive Technologies. Springer Berlin Heidelberg, 2008.

[GF09] Martin Gasser and Arthur Flexer. Fm4 soundpark audio-based music recommendation in everyday use. In *Proceedings of the 6th Sound and Music Computing Conference (SMC '09)*, pages 162–166, Porto, Portugal, Jul. 23-25 2009.

[GNOT92] David Goldberg, David Nichols, Brian M. Oki, and Douglas Terry. Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35(12):61–70, 1992.

[Got06] Masataka Goto. A chorus section detection method for musical audio signals and its application to a music listening station. *IEEE Transactions on Audio, Speech, and Language Processing*, 14(5):1783–1794, Sep. 2006.

[HBC09] Matthew D. Hoffman, David M. Blei, and Perry Cook. Easy as cba: A simple probabilistic model for tagging music. In *Proceedings of the 10 International Society for Music Information Retrieval Conference (ISMIR 09)*, pages 369–374, Kobe, Japan, Oct. 2009.

[Hla07] Peter Hlavac. Innovative Interfaces for Accessing Music on Mobile Devices. Master's thesis, Vienna University of Technology, Mar. 2007.

[IFP09] IFPI Digital Music Report 2009, Jan. 2009. `http://www.ifpi.org/content/library/DMR2009.pdf`.

[Koh95] Teuvo Kohonen. *Self-Organizing Maps*, volume 30 of *Springer Series in Information Sciences*. Springer, Berlin, Heidelberg, 1995.

[Lam08] Paul Lamere. Social tagging and music information retrieval. *Journal of New Music Research*, 37(2):101–114, 2008.

[Lid06] Thomas Lidy. Evaluation of New Audio Features and Their Utilization in Novel Music Retrieval Applications. Master's thesis, Vienna University of Technology, Dec. 2006.

[Log00] Beth Logan. Mel frequency cepstral coefficients for music modeling. In *Proceedings of the International Symposium on Music Information Retrieval (ISMIR)*, Plymouth, USA, Oct. 2000.

[Log02] Beth Logan. Content-Based Playlist Generation: Exploratory Experiments. In *Proceedings of the 3rd International Conference on Music Information Retrieval (ISMIR '02)*, Paris, France, Oct. 2002.

[LPA03] Amaury La Burthe, François Pachet, and Jean-Julien Aucouturier. Editorial Metadata in the Cuidado Music Browser: Between Universalism and Autism. In *Proceedings of the WedelMusic 2003 Conference*, pages 130–137, Leeds, U.K., Sep. 2003.

[LR05] Thomas Lidy and Andreas Rauber. Evaluation of Feature Extractors and Psycho-acoustic Transformations for Music Genre Classification. In *Proceedings of the Sixth International Conference on Music Information Retrieval (ISMIR 2005)*, pages 34–41, London, UK, Sep. 11-15 2005.

[LR06] Thomas Lidy and Andreas Rauber. Visually Profiling Radio Stations. In *Proceedings of the 7th International Conference on Music Information Retrieval (ISMIR '06)*, pages 186–191, Victoria, BC, Canada, Oct. 8-12 2006. University of Victoria.

[LR08] Thomas Lidy and Andreas Rauber. *Machine Learning Techniques for Multimedia*, chapter Classification and Clustering of Music for Novel Music Access Applications, pages 249–285. Cognitive Technologies. Springer, Berlin Heidelberg, Feb. 2008.

[LSSH09] Cyril Laurier, Mohamed Sordo, Joad Serrà, and Perfecto Herrera. Music mood representations from social tags. In *Proceedings of the 10 International Society for Music Information Retrieval Conference (ISMIR 09)*, pages 381–386, Kobe, Japan, Oct. 2009.

[LT07] Stefan Leitich and Markus Toth. PublicDJ - Music selection in public spaces as multiplayer game. In *Audio Mostly 2007*, pages 72–75, Ilmenau, Germany, Spe. 2007.

[LWM+09] Edith Law, Kris West, Michael Mandel, Mert Bay, and J. Stephen Downie. Evaluation of algorithms using games. In *Proceedings of the 10 International Society for Music Information Retrieval Conference (ISMIR 09)*, pages 387–392, Kobe, Japan, Oct. 2009.

[Mar04] José M. Martínez, editor. *MPEG-7 Overview (version 10). ISO/IEC JTC1/SC29/WG11N6828*. International Organisation for Standardisation, Palma de Mallorca, Spain, Sep. 2004.

[MLR06] Rudolf Mayer, Thomas Lidy, and Andreas Rauber. The map of mozart. In *Proceedings of the 7th International Conference on Music Information Retrieval (ISMIR'06)*, pages 351–352, Oct. 8-12 2006.

[MMR05] Rudolf Mayer, Dieter Merkl, and Andreas Rauber. Mnemonic SOMs: Recognizable shapes for self-organizing maps. In *Proceedings of the Fifth Workshop on Self-Organizing Maps (WSOM'05)*, pages 131–138, Paris, France, Sep. 5–8 2005.

[MUT+05] Fabian Mörchen, Alfred Ultsch, Michael Thies, Ingo Löhken, Mario Nöcker, Christian Stamm, Niko Efthymiou, and Martin Kümmerer. MusicMiner: Visualizing timbre distances of music as topographical maps. Technical report, Deptartment of Mathematics and Computer Science, University of Marburg, May 2005.

[MUTL06] Fabian Mörchen, Alfred Ultsch, Michael Thies, and Ingo Löhken. Modeling timbre distance with temporal statistics from polyphonic music. *IEEE Transactions on Audio, Speech and Language Processing*, 14(1):81–90, Jan. 2006.

[NDR05] Robert Neumayer, Michael Dittenbach, and Andreas Rauber. PlaySOM and PocketSOMPlayer: Alternative Interfaces to Large Music Collections. In *Proceedings of the Sixth International Conference on Music Information Retrieval (ISMIR 2005)*, pages 618–623, London, UK, Sep. 11-15 2005.

[NFH+07] Robert Neumayer, Jakob Frank, Peter Hlavac, Thomas Lidy, and Andreas Rauber. Bringing Mobile Map Based Access to Digital Audio to the End User. In *Proceedings of the 14th International Conference on Image Analysis and Processing (ICIAP 2007) and the Workshop on Visual and Multimedia Digital Libraries (VMDL07)*, pages 9–14, Modena, Italy, Sep. 2007. IEEE.

[Nil99] Martin Nilsson. ID3v2 informal standard. Technical report, ID3.org, Feb. 1999.

[OLJ+04] Kenton O'Hara, Matthew Lipson, Marcel Jansen, Axel Unger, Huw Jeffries, and Peter Macer. Jukola: democratic music choice in a public space. In *Proceedings of the 5th conference on Designing interactive systems (DIS '04)*, pages 145–154, New York, NY, USA, 2004. ACM.

[Ori06] Nicola Orio. Music retrieval: A tutorial and review. *Foundations and Trends in Information Retrieval*, 1(1):1–90, Sep. 2006.

[PVV06] Steffen Pauws, Wim Verhaegh, and Mark Vossen. Fast Generation of Optimal Music Playlists using Local Search. In *Proceedings of the 7th International Conference on Music Information Retrieval (ISMIR '06)*, pages 138–143, Victoria, BC, Canada, Oct. 8-12 2006. University of Victoria.

[PVV08] Steffen Pauws, Wim Verhaegh, and Mark Vossen. Music playlist generation by adapted simulated annealing. *Information Sciences*, 178(3):647–662, 2008. Including Special Issue "Ambient Intelligence".

[RF01] Andreas Rauber and Markus Frühwirth. Automatically Analyzing and Organizing Music Archives. In *Proceedings of the 5th European Conference on Research and Advanced Technology for Digital Libraries*

*(ECDL 2001)*, Springer Lecture Notes in Computer Science, Darmstadt, Germany, Sep. 4-8 2001. Springer.

[RPM03] Andreas Rauber, Elias Pampalk, and Dieter Merkl. The SOM-enhanced JukeBox: Organization and Visualization of Music Collections based on Perceptual Models. *Journal of New Music Research*, 32(2):193–210, Jun. 2003.

[She64] Roger N. Shepard. Circularity in Judgments of Relative Pitch. *The Journal of the Acoustical Society of America*, 36(12):2346–2353, 1964.

[SWV+08] Sanni Siltanen, Charles Woodward, Seppo Valli, Petri Honkamaa, Andreas Rauber, Jakob Frank, Thomas Lidy, and Robert Neumayer. Natural/ novel user interfaces for mobile devices. In *Multimodal Processing and Interaction - Audio, Video, Text*. Springer, 2008.

[TC99] George Tzanetakis and Perry Cook. Multifeature audio segmentation for browsing and annotation. In *Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA '99)*, pages 103–106, Oct. 1999.

[TC00a] George Tzanetakis and Perry Cook. MARSYAS: A Framework for Audio Analysis. *Organized Sound*, 4(30):169–175, 2000.

[TC00b] George Tzanetakis and Perry Cook. Sound analysis using MPEG compressed audio. In *Proceedings of the Acoustics, Speech, and Signal Processing (ICASSP '00)*, pages II761–II764, Washington, DC, USA, 2000. IEEE Computer Society.

[Tza02] George Tzanetakis. *Manipulation, Analysis and Retrieval Systems for Audio Signals*. PhD thesis, Computer Science Department, Princeton University, 2002.

[UM05] Alfred Ultsch and Fabian Mörchen. Esom-maps: tools for clustering, visualization, and classification with emergent som. Technical Report 46, Deptartment of Mathematics and Computer Science, University of Marburg, 2005.

[YKK+04] Takuya Yoshioka, Tetsuro Kitahara, Kazunori Komatani, Tetsuya Ogata, and Hiroshi G. Okuno. Automatic chord transcription with concurrent recognition of chord symbols and boundaries. In *Proceedings of the 5th*

*International Conference on Music Information Retrieval (ISMIR 2004)*, pages 100–105, Barcelona, Spain, Oct. 2004.

[Zwi61] E. Zwicker. Subdivision of the audible frequency range into critical bands. *The Journal of the Acoustical Society of America*, 33(2):248–248, 1961.

# A User Study

## A.1 Playlists

### A.1.1 Playlist 1 – last.fm (distributed)

| # | Title | Artist |
|---|---|---|
| 1 | Poolgaye | ADH |
| 2 | Choctaw Hayride | Alison Krauss & Union Station |
| 3 | Krzesany | Wojciech Kilar |
| 4 | Never the Same | Supreme Beings of Leisure |
| 5 | Alech Taadi | Khaled |
| 6 | Bayaty | Ashkhabad |
| 7 | Tu Mirá | Lole y Manuel |
| 8 | Malinche | Lila Downs |
| 9 | St. James Infirmary | Cab Calloway |
| 10 | Sprout And The Bean | Joanna Newsom |
| 11 | On & On | Erykah Badu |
| 12 | Mama, You Been On My Mind | Jeff Buckley |
| 13 | Wake Up | The Ditty Bops |

### A.1.2 Playlist 2 – last.fm (focused)

| # | Title | Artist |
|---|---|---|
| 1 | It Ain't Hard to Tell | Nas |
| 2 | Electric Relaxation | A Tribe Called Quest |
| 3 | Time Is the Enemy | Quantic |
| 4 | Epilogue | Kwan |
| 5 | Doo Wop (That Thing) | Lauryn Hill |
| 6 | Didn't Cha Know | Erykah Badu |
| 7 | Fu-Gee-La | Fugees |
| 8 | Luchini | Camp Lo |
| 9 | Could You Be Loved | Bob Marley |
| 10 | Lynguistics | CunninLynguists |
| 11 | Drums of Death | DJ Shadow |
| 12 | Quantic | Time Is The Enemy |
| 13 | Gatheration | Lady Sovereign |

### A.1.3 Playlist 3 – radio (focused)

| # | Title | Artist |
|---|---|---|
| 1 | Wisemen | James Blunt |
| 2 | You Sang to Me | Marc Anthony |
| 3 | Beautiful | Christina Aguilera |
| 4 | Run | Leona Lewis |
| 5 | Without You | Nilsson |
| 6 | '74-'75 | Connells |
| 7 | Love Remains the Same | Gavin Rossdale |
| 8 | Holding Back the Years | Simply Red |
| 9 | Halo | Beyoncé |
| 10 | Mandy | Westlife |
| 11 | Alone | Heart |
| 12 | Use Somebody | Kings of Leon |
| 13 | Spending My Time | Roxette |

### A.1.4 Playlist 4 – generated (distributed)

| # | Title | Artist |
|---|---|---|
| 1 | Wide Open Spaces | Dixie Chicks |
| 2 | The Void | Mother Tongue |
| 3 | Kick | White Rose Movement |
| 4 | Rooftops | Alkaline Trio |
| 5 | Happiness Is a Warm Gun | The Beatles |
| 6 | Washer | Slint |
| 7 | Metal Gods | Judas Priest |
| 8 | Tombstone | Suzanne Vega |
| 9 | San Francisco Knights | People Under the Stairs |
| 10 | Leave (Get Out) | JoJo |
| 11 | Diamonds From Sierra Leone (Remix) | Kanye West |
| 12 | Manana en el abasto | Sumo |
| 13 | No Way | The Books |

### A.1.5 Playlist 5 – generated (focused)

| # | Title | Artist |
|---|---|---|
| 1 | The Night Before the Day the Earth Stood Still | New Bomb Turks |
| 2 | Fuego Interno | Thermo |
| 3 | The Melting Point of Wax | Thrice |
| 4 | An Enemy Led the Tempest | Cradle of Filth |
| 5 | Take A Drink | Quietdrive |
| 6 | See You | Saves the Day |
| 7 | Poison Prince (Lo-Fi Acoustic Version) | Amy Macdonald |
| 8 | Ceremony (Converter Vs. Empusae) | Empusae |
| 9 | Allahi Allah (Carmen Rizzo Remix) | Niyaz |
| 10 | So Far Away | Nine Days |
| 11 | Red Sun | Thin White Rope |
| 12 | Above the Lone | Love Spirals Downwards |
| 13 | How Near, How Far | ...And You Will Know Us by the Trail of Dead |

### A.1.6 Playlist 6 – radio (distributed)

| # | Title | Artist |
|---|---|---|
| 1 | Ayo Technology | Milow |
| 2 | Africa | Toto |
| 3 | I Wanna be the Only One | Eternal |
| 4 | The End Of Days | My Excellence |
| 5 | Lasse redn | die ärzte |
| 6 | She Drives Me Crazy | Fine Young Cannibals |
| 7 | Wire to Wire | Razorlight |
| 8 | The Bad Touch | Bloodhound Gang |
| 9 | Die bessere Hälfte | Herbstrock |
| 10 | Sacrifice | Elton John |
| 11 | All About Us | t.A.T.u. |
| 12 | I'm on Fire | Bruce Springsteen |
| 13 | I Can See Clearly Now | Jimmy Cliff |

## A.2 Questionaire

The following is an exemaplary questionnaire (for *Playlist 2 – last.fm (focused)*) the participants were asked to fill out during the user study.

```
#2.m3u
This is the questionaire for 2.m3u

While listening to the playlist "2.m3u", please rate how convenient
you conceive the combination of two consecutive songs.

Please select your answer by picking the apropriate value from a
scale from 1 to 5 (including), where "5" stands for a very good
combination and "1" for a misfit:

 very good   good      okay     not good   misfit
    5         4         3         2         1

Please write your answers between the songs after the S:

Nas - It Ain't Hard to Tell
1347582.mp3

S:

A Tribe Called Quest - Electric Relaxation
```

```
1064689.mp3

S:

Quantic - Time Is the Enemy
1033309.mp3

S:

Kwan - Epilogue
3335512.mp3

S:

Lauryn Hill - Doo Wop (That Thing)
17222.mp3

S:

Erykah Badu - Didn't Cha Know
1433571.mp3

S:

Fugees - Fu-Gee-La
1014274.mp3

S:

Camp Lo - Luchini
1214357.mp3

S:

Bob Marley - Could You Be Loved
1069673.mp3

S:

CunninLynguists - Lynguistics
1753517.mp3

S:

DJ Shadow - Drums of Death
1647936.mp3
```

```
S:

Time Is The Enemy - Quantic
4127021.mp3

S:

Lady Sovereign - Gatheration
56891260.mp3


After you listened to the entire playlist some final questions:

On a scale from 1 to 5 (where 5 is the best): What is your overall
rating of this playlist?


For which situations would you select this playlist? (e.g. party,
car, candlelight dinner, ...)


If you should remove up to three songs from this list to improve the
overall atmosphere of this playlist, which do you select?


And if you want to leave some further comments on this playlist,
please add them here:



#EOF
```

# A.3 Answers

## A.3.1 Playlist 1 – last.fm (distributed)

| Participant | A | B | C | D | E |
|---|---|---|---|---|---|
| 1–2 | 1 | 2 | 2 | 3 | 2 |
| 2–3 | 1 | 1 | 1 | 1 | 1 |
| 3–4 | 1 | 1 | 1 | 1 | 1 |
| 4–5 | 2 | 3 | 1 | 2 | 3 |
| 5–6 | 2 | 3 | 4 | 3 | 5 |
| 6–7 | 1 | 3 | 2 | 2 | 4 |
| 7–8 | 3 | 4 | 5 | 2 | 5 |
| 8–9 | 1 | 1 | 1 | 1 | 2 |
| 9–10 | 1 | 1 | 3 | 2 | 2 |
| 10–11 | 3 | 2 | 4 | 4 | 2 |
| 11–12 | 2 | 1 | 2 | 3 | 3 |
| 12–13 | 2 | 2 | 1 | 2 | 2 |
| Overall Rating | 1 | 3 | 2 | 2 | 2 |
| Songs to remove | 13, 11, 8 | 10, 3, 9 | 3, 9, 12 | 3, 9, 2 | 3, 9 |

| Part. | Situation | Comment |
|---|---|---|
| A | no way that i would select that! | |
| B | driving | |
| C | Zur Entspannung -> car? | |
| D | Relaxing or calming down | |
| E | car, ironing, cooking, coffee | stilbruch passiert öfter als bruch in der “lieder-Laune” |

### A.3.2 Playlist 2 – last.fm (focused)

| Participant | A | B | C | D | E |
|---|---|---|---|---|---|
| 1—2 | 3 | 5 | 5 | 4 | 5 |
| 2—3 | 2 | 3 | 1 | 3 | 3 |
| 3—4 | 2 | 5 | 2 | 3 | 4 |
| 4—5 | 1 | 2 | 3 | 2 | 4 |
| 5—6 | 1 | 2 | 3 | 3 | 3 |
| 6—7 | 2 | 3 | 2 | 3 | 4 |
| 7—8 | 3 | 4 | 4 | 4 | 5 |
| 8—9 | 1 | 2 | 2 | 4 | 3 |
| 9—10 | 3 | 3 | 2 | 3 | 3 |
| 10—11 | 3 | 2 | 3 | 2 | 4 |
| 11—12 | 3 | 3 | 3 | 4 | 3 |
| 12—13 | 2 | 1 | 3 | 1 | 3 |
| Overall Rating | 3 | 1 | 3 | 3 | 4 |
| Songs to remove | 5, 13, 10 | | 3, 9, 12 | 13 | 9, 12 |

| Part. | Situation | Comment |
|---|---|---|
| A | not my taste; i won't select it; | |
| B | I wouldn't want to listen to most of the music in the playlist. Situation? Volume turned down all the way? | Three is not enough ;-) |
| C | party | |
| D | Hanging out with friends | Duplicate entry of Quantic/Time is the enemy... Song transitions not too bad, but switching genres back and forth irritates (RnB->Electro/Chillout->RnB->Reaggea->Electro->Black) |
| E | party, car, coffee | |

### A.3.3 Playlist 3 – radio (focused)

| Participant | A | B | C | D | E |
|---|---|---|---|---|---|
| 1–2 | 4 | 3 | 4 | 3 | 5 |
| 2–3 | 4 | 4 | 4 | 2 | 4 |
| 3–4 | 5 | 3 | 3 | 4 | 5 |
| 4–5 | 4 | 4 | 5 | 3 | 5 |
| 5–6 | 3 | 3 | 3 | 2 | 4 |
| 6–7 | 3 | 3 | 3 | 2 | 4 |
| 7–8 | 2 | 3 | 3 | 2 | 4 |
| 8–9 | 3 | 2 | 2 | 3 | 3 |
| 9–10 | 3 | 2 | 4 | 2 | 3 |
| 10–11 | 4 | 3 | 3 | 4 | 5 |
| 11–12 | 2 | 3 | 3 | 4 | 5 |
| 12–13 | 3 | 3 | 4 | 3 | 5 |
| Overall Rating | 4 | 3 | 4 | 2 | 5 |
| Songs to remove | 9, 4, 10 | 2, 4, 5 | 8, 6, 11 | 2, 6, 7 | 9 |

| Part. | Situation | Comment |
|---|---|---|
| A | car | |
| B | I wouldn't select it if there were other music to listen to. But if there only is this one radio station (with the best of the 70ies, 80ies, 90ies, and the hits of today %-)): Background music for other activities were you don't really listen (working, reading, smalltalk) | Don't we all love Oe3? Not. (That's why its playlist #3, isn't it?) |
| C | lovesickness | |
| D | Candlelight dinner / Romantic evening | You produced this playlist only on the snippets, right? Otherwise the Kings of Leon would be rather strange ;-) |
| E | candlelight dinner | teilweise habe ich nicht mitbekommen, das es schon das nächste lied ist |

### A.3.4 Playlist 4 – generated (distributed)

| Participant | A | B | C | D | E |
|---|---|---|---|---|---|
| 1–2 | 1 | 3 | 2 | 3 | 4 |
| 2–3 | 1 | 2 | 3 | 2 | 4 |
| 3–4 | 3 | 3 | 3 | 1 | 3 |
| 4–5 | 1 | 3 | 2 | 1 | 2 |
| 5–6 | 3 | 2 | 4 | 3 | 5 |
| 6–7 | 2 | 2 | 3 | 3 | 1 |
| 7–8 | 1 | 1 | 1 | 2 | 1 |
| 8–9 | 1 | 2 | 1 | 1 | 4 |
| 9–10 | 2 | 2 | 2 | 1 | 3 |
| 10–11 | 1 | 2 | 2 | 1 | 3 |
| 11–12 | 2 | 2 | 2 | 1 | 2 |
| 12–13 | 2 | 1 | 1 | 1 | 1 |
| Overall Rating | 2 | 3 | 2 | 2 | 2 |
| Songs to remove | 2, 13, 7 | 3, 9, 10 | 8, 10, 7 | 4 | 7, 12 |

| Part. | Situation | Comment |
|---|---|---|
| A | i won't select it | |
| B | Walking, public transport | The songs are nice enough but the combination is pretty weird. (Is this really the FM4 playlist? ;-)) |
| C | car | |
| D | Listening to different styles of music (really, I don't get the idea of the list) | Beatles - Slint - Judas Priest is a nice progression |
| E | party, car | |

### A.3.5 Playlist 5 – generated (focused)

| Participant | A | B | C | D | E |
|---|---|---|---|---|---|
| 1–2 | 3 | 4 | 3 | 4 | 4 |
| 2–3 | 3 | 5 | 4 | 4 | 5 |
| 3–4 | 2 | 3 | 3 | 4 | 2 |
| 4–5 | 2 | 3 | 1 | 3 | 2 |
| 5–6 | 3 | 5 | 4 | 4 | 5 |
| 6–7 | 1 | 4 | 2 | 1 | 3 |
| 7–8 | 2 | 2 | 1 | 1 | 1 |
| 8–9 | 3 | 4 | 3 | 1 | 5 |
| 9–10 | 3 | 2 | 2 | 1 | 2 |
| 10–11 | 3 | 4 | 4 | 3 | 4 |
| 11–12 | 2 | 3 | 2 | 1 | 4 |
| 12–13 | 2 | 3 | 3 | 1 | 3 |
| Overall Rating | 3 | 4 | 2 | 3 | 4 |
| Songs to remove | 9, 12, 8 | 1, 4, 9 | 4, 8, 9 | 7, 8, 9 | 4, 8, 9 |

| Part. | Situation | Comment |
|---|---|---|
| A | party | |
| B | party, beach, driving, hanging out | |
| C | during sport | |
| D | Preparing for going out (at least the first part) | What happened halfway through? |
| E | party, car | |

### A.3.6 Playlist 6 – radio (distributed)

| Participant | A | B | C | D | E |
|---|---|---|---|---|---|
| 1–2 | 4 | 2 | 3 | 1 | 5 |
| 2–3 | 3 | 4 | 4 | 4 | 4 |
| 3–4 | 5 | 2 | 2 | 2 | 3 |
| 4–5 | 3 | 2 | 1 | 1 | 2 |
| 5–6 | 2 | 2 | 2 | 1 | 2 |
| 6–7 | 3 | 2 | 3 | 3 | 3 |
| 7–8 | 3 | 2 | 2 | 1 | 3 |
| 8–9 | 4 | 2 | 3 | 1 | 3 |
| 9–10 | 4 | 3 | 3 | 2 | 2 |
| 10–11 | 1 | 2 | 3 | 1 | 2 |
| 11–12 | 3 | 4 | 2 | 1 | 2 |
| 12–13 | 3 | 3 | 2 | 2 | 4 |
| Overall Rating | 4 | 4 | 4 | 1 | 3 |
| Songs to remove | 13, 1, 5 | 2, 3, 10 | 5, 10, 12 | | |

| Part. | Situation | Comment |
|---|---|---|
| A | car | |
| B | hanging out | (Please, no more obnoxious earworms!!!) Again: Most songs are nice but the combination isn't very good. This would make 2 good playlists if divided into calmer and more "active" songs. |
| C | car, relax | |
| D | No idea - listening to different music? | This isn't shuffle on a big collection?? |
| E | party, car | schwierig, eher breiter gefächert, man kann nicht nur 3 ausortieren, weil es keine eindeutig störenden lieder gibt |