DIPLOMARBEIT

Islands of Music

Analysis, Organization, and Visualization of Music Archives

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

unter der Anleitung von ao.Univ.Prof. Dr. Dieter Merkl
und Univ.Ass. Dr. Andreas Rauber
als verantwortlich mitwirkendem Universitätsassistenten

durch

Elias Pampalk
Absberggasse 25/2/11
1100 Wien

Matr.Nr. 9625173

Wien, Dezember 2001

Zusammenfassung

Inseln der Musik sind eine graphische Benutzerschnittstelle zu Musiksammlungen basierend auf einer Metapher von geographischen Landkarten. Inseln repräsentieren Musikgenres. Berge und Hügel auf diesen Inseln repräsentieren Subgenres. Die Inseln sind derart angeordnet, dass ähnliche Genres nahe beisammen und vielleicht sogar durch eine Landpassage verbunden sind, während Genres, welche als sehr unterschiedlich wahrgenommen werden, durch ein tiefes Meer getrennt sind. Die Musikstücke aus der Sammlung werden auf der Karte entsprechend ihrem Genre platziert. Um eine Navigation auf der Karte zu unterstützen sind die Berge und Hügel mit Bezeichnungen versehen welche rhythmischen und anderen Eigenschaften der repräsentierten Genres entsprechen.

Inseln der Musik sind gedacht, um die Exploration von unbekannten Musiksammlungen zu unterstützen. Sie könnten von Musikgeschäften eingesetzt werden, um ihren Kunden bei der Suche nach etwas Neuem zu helfen, genauso wie sie als Schnittstelle zu digitalen Musikbibliotheken dienen können, oder einfach zur Organisation der Musiksammlung zu Hause.

Diese Diplomarbeit behandelt jene Herausforderungen welche bei der automatischen Erzeugung solcher Inseln der Musik auftreten, wenn lediglich reine Musikdaten (z.B. MP3s) gegeben sind ohne zusätzlicher Information, wie zum Beispiel die Zuordnung der Musikstücke zu Genres. Die größte Herausforderung liegt darin, die wahrgenommene Ähnlichkeit zweier Musikstücke zu berechnen. Basierend auf Modelle der Psychoakustik wird ein Ansatz präsentiert mit Fokus auf den dynamischen Eigenschaften der Musik. Ein Neuronales Netz, im speziellen die Self-Organizing Map, wird verwendet, um die Musiksammlung zu organisieren. Mittels einer neuen Visualisierungstechnik wird die Karte mit den Inseln erzeugt. Des weiteren werden Methoden zur Beschreibungen der Berge und Hügel vorgestellt.

Abstract

Islands of Music are a graphical user interface to music collections based on a metaphor of geographic maps. Islands represent musical genres. Mountains and hills on these islands represent sub-genres. The islands are situated in such a way that similar genres are close together and might even be connected by a land passage while perceptually very different genres are separated by deep sea. The pieces of music from the collection are placed on the map according to their genre. To support navigation on the map the mountains and hills are labeled with words which describe rhythmic and other properties of the genres they represent.

Islands of Music are intended to support exploration of unknown music collections. They could be utilized by music stores to assist their customers in finding something new to buy. They could also serve as interfaces to digital music libraries, or they could simply be used to organize one's personal music collection at home.

This thesis deals with the challenges involved in the automatic creation of such Islands of Music given only raw music data (e.g. MP3s) without any further information such as which genres the pieces of music belong to. The main challenge is to calculate the perceived similarity of two pieces of music. An approach based on psychoacoustics is presented which focuses on the dynamic properties of music. Using a neural network algorithm, namely the self-organizing map, the music collection is organized and a novel visualization technique is facilitated to create the map of islands. Furthermore, methods to find descriptions for the mountains and hills are demonstrated.

Contents

1. Introduction

Music is magic. It influences our emotions. It has the power to make us happy or sad, just as it can make us relaxed or aggressive. Often it is associated with some very special moments in our lives. Moreover, music is an important part of our cultures and identities. However, the most fascinating aspect of music might be the fact that the annual turnover for record sales only within the US has a magnitude of several billion USD.

This huge industry would not exist without its customers, who are always looking for something new to listen to. There are many ways in which customers find their desired products.

For example, one way is to listen around. Customers might listen to what is being played on the radio or to what friends are listening to. However, this type of search is restricted to the subjective taste of others. Furthermore, it might take a while until a new release reaches ones ears.

Another approach is to follow the development of artists, who have been appreciated in the past, assuming that their work will also be appreciated in the future. However, this kind of search does not include unknown artists or newcomers.

Customers might also follow the development of a genre like Jazz, Hip Hop, Classic or Funk. Relying on the classification skills of other people it is possible to search music stores for new releases in certain genres. However, classifying music into a limited number of genres is not an easy task. Lots of music is located somewhere in between many genres.

The tool presented in this thesis is meant to help customers find music without limiting the search to a specific genre or artist. This tool is based on a metaphor of geographic maps. Genres of music are represented by islands and continents. Similar genres are located close together and might even be connected through some land passage. On these islands there might be some further sub-genres that are represented as mountains and hills. Again these sub-genres might be more or less similar to each other and are arranged accordingly.

The mountains and hills on the map are labeled with words that describe certain attributes of the associated genre, for example the type of rhythm is described rather than using words like Pop, Jazz or Classical.

The pieces of music in the collection are placed on the map according to their genre or sub-genre. Most of the music will be located around the mountains. However the few located in the valleys between typical genres might be the most interesting ones.

The user can listen to the music by clicking on its representation on the map and can explore island after island according to his or her musical taste. Furthermore, music known to the user can be used as landmarks, to identify interesting regions on the map.

The maps with the islands of music could easily be placed on a web page of an

internet store. Or they could be used in any conventional music store. Simple earphones and a touch screen monitor connected to a server would be sufficient.

These maps could also be applied to digital libraries containing music, or simply at home to organize the private music collection. They could reveal some interesting properties of the inherent structure of the music collection that might not have been obvious before.

The technical requirements to develop music maps have only recently been met by the tremendous increase in computational power as well as the availability of affordable large storage. Now it is possible to handle the huge amount of data within music collections and to do the complex calculations leading to the music maps described above within reasonable time.

1.1. Scope and Overview

This thesis explores two main aspects related to music maps. One is how to compute the similarity of two pieces of music, so a whole music collection can be organized accordingly. The second aspect is how to present this information to the user in an intuitive way. The main goal of this thesis is to demonstrate the possibility of building a system, which enables efficient exploration of unknown music collections, given only the raw pieces of music without any meta information.

This thesis uses a music collection of 359 pieces with a total length of about 23 hours to illustrate and evaluate the methods. A detailed list of all pieces of music, their authors and titles can be found in Appendix B. The music collection consists of a mixture of pieces of music from different genres. Most of these pieces are well known so that the reader can easily verify the presented results.

Related work can be found in *Chapter 2*. In particular the fields of content-based music analysis and approaches based on the Self-Organizing Map are discussed. Work related to details on psychoacoustics, clustering algorithms, and the visualization and automatic summarization of the results is presented in the corresponding chapters.

In *Chapter 3* the methods used to extract relevant features, which enable the computer to compare two pieces of music, are presented. These features are derived from the low-level raw audio signal without any additional meta information. Based on psychoacoustic findings, features are constructed which reflect the dynamic and rhythmic properties of music. All feature extraction steps are illustrated using pieces of music from the music collection.

Chapter 4 deals with different approaches used to combine and analyze the extracted features. Four different approaches to describe pieces of music with different lengths are presented. All methods are evaluated with the music collection.

The simplest of these approaches, which produces a quality that is comparable to the other approaches, is used to describe a possible user interface in *Chapter 5*. A novel method to visualize clusters in a Self-Organizing Map is presented along with

methods to give automatic summaries for groups of music. Furthermore, a HTML-based demonstration with a small subset of the music collection, which is available on the internet[1], is briefly described. This demonstration is intended to enable the reader to explore and to listen to Islands of Music.

Finally in *Chapter 6* this work is concluded. A summary and further work together with interesting directions, with possibly very promising results are discussed.

1.2. Notation and Conventions

In the mathematical notation an italic typeface indicates scalar values, e.g. f, t. The indexes of matrices are usually indicated by i, j, k and if not stated otherwise, run from the lowest to the highest index. Using the Matlab® convention all indexes of vectors and matrices start with 1. Bold typeface indicates vectors and matrices, the former using lower case symbols, e.g. $\mathbf{x}, \mathbf{m}$, and the latter using upper case symbols, e.g. $\mathbf{P}, \mathbf{S}$. Note however, that exceptions to this convention do appear.

[1] `http://student.ifs.tuwien.ac.at/~elias/music`

2. Related Work

Music has been analyzed since the ancient Greeks. Pythagoras is credited with recognizing that strings whose lengths are related as the ratio of small integers sounds good when plucked at the same time. Since then a lot of research has been conducted and very sophisticated models and systems have been developed.

In the scope of this thesis especially systems which are designed to search for music based on its content are interesting, since this is the main motivation for this thesis. Section 2.1 reviews the literature on content-based music retrieval and section 2.2 focuses on approaches using the SOM algorithm. Work related to details on psychoacoustics, clustering algorithms, and the visualization and automatic summarization of the results is presented in the respective chapters.

2.1. Content-Based Music Retrieval

There are several possibilities to search for music based on its content. One is to use metadata information consisting of descriptions which have manually been assigned to each piece of music. These descriptions can be as simple as the name of the piece of music, but also more complex like an assignment to a specific genre or a verbal description of the music. The necessary standards are provided, for example, by the MPEG 7 standard [NL99]. A system based on MPEG 7 to compare sounds has, for example, been presented by [PMH00].

Often pieces of music have lyrics, thus another possibility would be to apply methods from text document retrieval to search for music. Such systems are currently in use, for example, *BigLyrics.com* and *LyricCrawler.com* are two of the currently biggest music lyric search engines on the web. Using the lyrics as descriptions of the music it is possible to create an interface to allow an exploration of music archives where pieces of music with similar lyrics are located close to each other on a 2-dimensional map display using methods which have been developed mainly for text document collections, such as the SOMLib [RM99] or the WebSOM [KHLK98].

Sometimes music is available in the MIDI format [MID96] and for a limited subset of music automatic transcription to the MIDI format is possible (e.g. [Moo77, MSW96b]). Music archives in MIDI format contain information on the exact melody for each song. Hawley presented a system [Haw90], where the user enters a melody on a keyboard and tunes whose beginnings exactly match the input are retrieved. Ghias et al. [GLCS95] presented a system where the user hummes a query, which is reduced to relative information such as if a note is higher, lower or about the same as the previous, and retrieves songs which have similar melodies. One of the best known representatives of such systems is the New Zealand Musical Digital Library [MSW+96a, BNMW+99].

Additionally to the melody of a piece of music information on its style can be

abstracted from the MIDI data. Dannenberg *et al.* [DTW97] described a system, which classified solo improvised trumpet performances into one of the four styles: *lyrical*, *frantic*, *syncopated*, or *pointillistic*.

However, not always metadata is available and the metadata available might be erroneous, incomplete or inaccurate due to the deficiencies of manual labor. Likewise song lyrics are not always available, speech recognition systems only have a limited capability of extracting the lyrics automatically. And finally most music is available in MP3 or other similar formats rather than in MIDI. Thus content-based systems which directly analyze the raw music data (acoustical signals) have been developed. The models of these systems for music processing are usually sub-symbolic [Lem89] since they operate directly on acoustical signals and do not involve an explicit symbolization step as, for example, in [Wid01], where the musical structure is analyzed on a very abstract level.

An overview of systems analyzing audio databases was presented by Foote [Foo99]. However, Foote focuses particularly on systems for retrieval of speech or partly-speech audio data. Several studies and overviews related to content-based audio signal classification are available (e.g. [Ger00, LW01]), however, they do not treat content-based music classification in detail.

Wold *et al.* [WBKW96] presented a system which analyzes sounds based on their pitch, loudness, brightness, and bandwidth over time and tracked the mean, variance, and autocorrelation functions of these properties. They analyze sounds such as speech and individual musical notes, but do not focus on whole music collections.

Other approaches (e.g. [Foo97, BKWW99, Log00]) are based on methods developed in the digital speech processing community using *Mel Frequency Cepstral Coefficients* (MFCCs). MFCCs are motivated by perceptual and computational considerations, for example, instead of calculating the exact loudness sensation only decibel values are used. Furthermore the techniques appropriate to process speech data are not necessarily the best for processing music. For example, the MFCCs ignore some of the dynamic aspects of music.

Recently Scheirer [Sch00] presented a model of human perceptual behavior and briefly discussed how his model can be applied to classifying music into genre categories and performing music similarity-matching. However, he has not applied his model to large scale music collections. The collection he uses consisted of 75 songs from each of which he selected two 5-second sequences.

2.2. Approaches using Self-Organizing Maps

This thesis is built upon the work of Frühwirth and Rauber [Frü01, FR01, RF01], who have shown that it is possible to cluster and organize music using neural networks. In their work they extract features from MP3 files which enable a self-organizing map (SOM) [Koh01] to learn the inherent structure within a music collection. The feature

extraction process consists of several steps.

They first transform the audio signals into the frequency domain using a fast Fourier transformation (FFT) with about 20 millisecond windows. In the frequency domain they select 17 frequencies for further processing. They split each piece of music into 5-second sequences. They remove the first and the last sequences to avoid fade-in and fade-out effects. From the remaining they select a subset using only every second to third sequence. Each frequency band from the selected sequences is then transformed into the frequency domain yielding 256 coefficients. They combine these 256 values for the 17 bands in a 4352-dimensional vector representing a 5-second sequence. These vectors reflect the dynamic properties of the selected frequencies.

A SOM is used to organize the large number of 5 second sequences on a 2-dimensional map in such a way that similar sequences are located close to each other. The different sequences of one piece of music might be scattered across the map if it contains a lot of variations. To get the pieces together again another feature vector is created using the information on where the different sequences of one piece of music are located. With this information another SOM is trained which organizes the pieces of music on a 2-dimensional map.

This thesis follows some of the proposals for further work presented by Frühwirth [Frü01] and tries to combine the well working approach with psychoacoustics methods to improve the performance.

An overview of psychoacoustics can be found in [ZF99]. Psychoacoustics deals with the relationship of physical sounds and the human brain's interpretation of them. For example, Feiten [FG94] applies psychoacoustics to cluster about 100 different music instruments according to their sounds. He uses the concepts of critical-bands and specific loudness, which are further discussed in Chapter 3. Feiten divides the sounds into sequences of very short (6 milliseconds) steady-state components, assuming that similar sounds are similar sequences of steady-state sounds. He then applies the SOM algorithm to order them according to their similarity on a 2-dimensional map. Each dynamic sound is represented by a trajectory on this map, and similar dynamic sounds have similar trajectories, which Feiten uses to create a (SOM) map of the music instruments.

Feiten's approach is very accurate and works very well with music instruments. However it is computationally expensive. Using only 300 millisecond sounds to describe each of the 100 music instruments, yields 5000 different steady-state sounds. A small music collection with about 10 hours of music would result in about 6 million steady-state sounds.

A similar approach as Feiten's where a SOM was used to cluster music instruments was presented by Cosi *et al.* [CPL94]. While Feiten applies general psychoacoustic models, Cosi *et al.* rely on specific methods developed by the speech processing community to process the audio signals.

Another approach using the SOM was published by Leman [Lem94, Lem95]. Leman used the SOM to represent tonality in music and using this 2-dimensional representation

to analyze cadence sequences of pieces of music. Leman's goal is to analyze music and he is especially interested in musicology and hopes to provide tools of interest for music theorists in understanding the nature of music. Thus the goal is rather different to the one of this thesis.

3. Feature Extraction

Music with a duration of 5 minutes is usually represented by 13 million values. These values describe the physical properties of the acoustical waves, which we hear. When analyzing this data it is necessary to remove the irrelevant parts and emphasize the important features. The extraction of these features from the raw data is the most critical part in the process of creating a content-based organization in a music collection. If it were possible to extract one single feature that directly indicates which genre a piece of music belongs to, everything else would be trivial.

Good features should be intuitively meaningful, based on psychoacoustic findings, and robust towards variations which are insignificant to our hearing sensation. Furthermore, they should lead to an organization of the music collection that makes sense and not be too expensive to compute.

It is necessary to consider computational aspects because the raw data of even small music collections easily consumes several gigabytes of storage. A detailed analysis of all this information and all its possible meanings would be computationally prohibitive. It thus is necessary to reduce the amount of information to what is relevant in respect to the overall goal, which is to organize music according to its genre. These genres are not clearly defined and different people might assign the same piece of music to different genres. However, there are some attributes of the raw data, which definitely do not determine the genre. For example, removing the first second of a piece of music does not change its genre, but the raw data compared bit wise will be completely different. Generally the duration of a piece of music is not relevant. Neither does a particular melody define a genre. The same melody can be interpreted in different genre styles just as different melodies might be members of the same genre. Likewise, the number of instruments involved plays a minor role in defining the genre.

One of the attributes that is rather typical for a genre is its rhythm which is why this thesis primarily focuses on the dynamics of music, and in particular on the fluctuation strength [Fas82] of the specific loudness per critical-band [Fel55, ZFS57].

The following sections describe the feature extraction steps starting with the raw data, which is transformed from the time-domain to its frequency-domain representation. In the frequency-domain several transformations are applied to obtain the specific loudness per critical-band.

Based on the specific loudness per critical-band the loudness fluctuation in a time interval of about 6 seconds is analyzed and an image in the dimensions of critical-band, modulation frequency, and fluctuation strength is created. To this image gradient and Gaussian filters are applied to emphasize important characteristics and remove insignificant ones. The modified fluctuation strength is used as final feature for the clustering algorithms.

3.1. Raw Audio Data

There are different possibilities to code digitized music. This thesis focuses on *Pulse Coded Modulation* (PCM) audio data to which any digital music representation can be transformed. For example, MP3 files can easily be decoded to PCM using almost any of the currently available audio players.

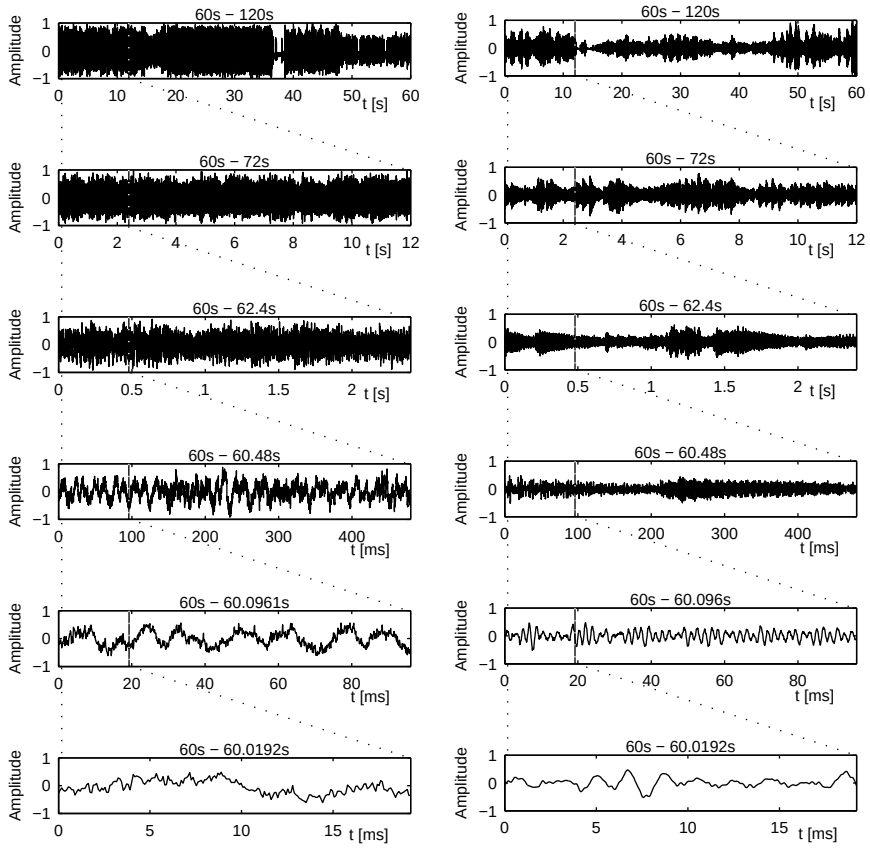
The PCM data is the discrete representation of a continuous acoustical wave. The amplitude is usually represented by 16 bits, which allows the description of more than 65,000 different amplitude levels. The time axis is usually sampled 44,100 times per second, which is one amplitude value approximately every 23 microseconds. The unit of the sampling frequency is *Hertz* (Hz), measured in cycles per second. The amplitude values are dimensionless although they correspond to sound pressure levels. The actual level depends on the sound system used to create physical acoustic waves.

Figure 3.1 illustrates 44kHz PCM data at different magnitude levels. The song *Freak on a Leash* by *Korn* is rather aggressive, loud, and perhaps even a little noisy. At least one electric guitar, a strange sounding male voice, and drums create a freaky sound experience. In contrast *Für Elise* by *Beethoven* is a rather peaceful, classical piano solo. These two pieces of music will be used throughout the thesis to illustrate the different steps of the feature extraction process. The envelope of the acoustical wave in *Für Elise* over the interval of one-minute has several modulations and seldom reaches the highest levels, while *Freak on a Leash* is constantly around the maximum amplitude level. There is also a significant difference in the interval of only 20 milliseconds (ms), where *Freak on a Leash* has a more jagged structure than *Für Elise*.

Obviously some basic descriptions of the audio signals would be sufficient to distinguish the peaceful-classical from the aggressive-noisy. One commonly used feature of signals is their zero-crossings (e.g. [SMK98, ZK98]), counting the times the signal crosses over the zero line. Another possibility is to analyze the envelope of the signal and extract beat information, based on the correlation between the magnitude of the envelope and the loudness [Dix00]. The main advantage of any features extracted directly from the PCM data is that they avoid computationally expensive preprocessing, which is necessary to transform the PCM data into data more representative of what we perceive. For example, it is not possible to recognize if there are vocals or not, or if the energy in the audio signal derives from instruments with high or low pitch.

Besides finding a representation that corresponds with our perception, it is also necessary to reduce the amount of data. A 5-second stereo sound sampled at 44kHz with 16 bit values equals almost 7 megabytes of data. For this reason the music is down-sampled to 11kHz, the two stereo channels are added up to one (mono), and only a fraction of every song is used for further processing. Specifically only every third 6-second sequence is further processed, starting after the first 12 seconds (fade-in) and ending before the last 12 seconds (fade-out). Additionally, zeros at the beginning or end of the music are truncated. This leads to a data reduction by a factor of over 16.

Changing stereo to mono has no significant impact on the perception of the music



(a) Korn, Freak on the Leash (b) Beethoven, Für Elise

Figure 3.1: The 44kHz PCM data of two very different pieces of music at different time scales. The titles of the subplots indicate their time intervals. The second minute was chosen since the first minute includes fade-in effects. Starting with an interval of 60 seconds each subsequent subplot magnifies the first fifth of the previous interval. This is indicated by the dotted lines. For each of the two pieces of music, the amplitude values are relative to the highest amplitude within the one-minute intervals.

genre. Using only a small fraction of an entire piece of music should be sufficient since humans are able to recognize the genre within seconds. Often it is possible to recognize the genre by listening to only one 6-second sequence of a piece of music. A more accurate classification is possible if a few sequences throughout the piece of music are listened to. However, the first and the last seconds of a piece of music usually contain fade-in and fade-out effects, which do not help in determining the genre. Neither should the down-sampling affect the ability to recognize the genre. In simple experiments using average computer speakers it was hardly possible to recognize the difference between 44kHz and 11kHz for most songs, while the genres remain clearly recognizable. Figure 3.2 depicts the effect of down-sampling. Notice that some of the fine details are lost, however, the signals still look alike. It is important to mention that down-sampling to 11kHz means that only frequencies up to 5.5kHz are noticeable, which will be discussed in the next section. This is definitely far below the 16kHz an average human can hear, however, 5.5kHz are sufficient to cover the spectrum we use in speech, and almost all

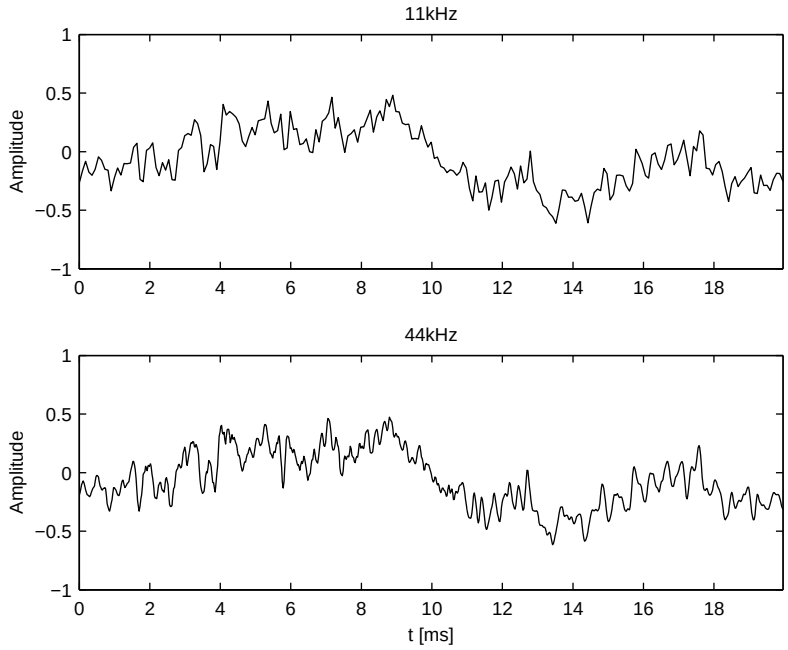


Figure 3.2: The effects of down-sampling on a 20ms sequence of *Freak on a Leash*. This sequence is the same as the one in the last subplot of Figure 3.1(a).

frequencies used in music [ZF99]. Furthermore, very high frequencies are usually not perceived as pleasant and thus do not play a very significant role in music.

3.2. Loudness Sensation

Loudness belongs to the category of intensity sensations. The loudness of a sound is measured by comparing it to a reference sound. The 1kHz tone is a very popular reference tone in psychoacoustics, and the loudness of the 1kHz tone at 40dB is defined to be 1 *sone*. A sound perceived to be twice as loud is defined to be 2 sone and so on.

To calculate the loudness sensation from raw audio data several transformations are necessary. The raw audio data is first decomposed into its frequencies using a discrete Fourier transformation. These frequencies are bundled according to the non-linear critical-band rate scale (bark). Then spectral masking effects are applied before the decibel values are calculated. The decibel values are transformed to equal loudness levels (phon) and finally from these the specific loudness sensation is calculated (sone).

At the end of this section each transformation is illustrated using examples from the music collection used for the experiments conducted for this thesis.

3.2.1. Discrete Fourier Transformation

Complex acoustical signals consist of several waves with different frequencies and amplitudes. The inner ear *(cochlea)* of humans decomposes the incoming acoustical waves into separate frequencies. The energy of different frequencies is transferred to and con-

centrated at different locations along the basilar membrane. Thus, it is appropriate to transform the PCM data into the frequency domain before analyzing it further. This can be achieved using, for example, *Fourier Transformations*. Alternatives include *Wavelets* [Chu92], but are not considered in this thesis.

Theory

In this subsection only the most important characteristics are summarized. A more detailed description can be found, for example, in [Bri74, PTVF92].

Using the linear Fourier transform, a continuous signal can be transformed between its time domain representation, denoted by $h(t)$, and the frequency domain representation $H(f)$. The respective equations are

$$\begin{aligned} H(f) &= \int_{-\infty}^{+\infty} h(t) e^{2\pi i f t} \mathrm{d}t, \quad \text{and} \\ h(t) &= \int_{-\infty}^{+\infty} H(f) e^{-2\pi i f t} \mathrm{d}f. \end{aligned} \tag{3.1}$$

Note that $i^2 = -1$ and $e^{ix} = \cos(x) + i \sin(x)$.

Pharseval's theorem, states that the total power in the signal is the same whether calculated in the time domain or the frequency domain. This is important for the loudness calculation of a sound and is formulated as

$$\int_{-\infty}^{+\infty} |h(t)|^2 \mathrm{d}t = \int_{-\infty}^{+\infty} |H(f)|^2 \mathrm{d}f. \tag{3.2}$$

The *one-sided power spectral density* $P_h(f)$, defines how much power is contained within the frequency interval between f and $f + \mathrm{d}f$. For real valued signals it is calculated as $P_h(f) = 2|H(f)|^2$.

As mentioned before, the audio signal is sampled at a fixed sampling rate, so the function is not continuous $h(t)$ but discrete $h(k\Delta t)$ with k denoting the whole-numbered index of the sample. The constant Δt denotes the time interval between two samples, which is determined by the sampling frequency (for 11kHz data Δt is less then 0.1ms).

For any sampling interval Δt there is a special frequency f_c, called the *Nyquist critical frequency*, given by $f_c \equiv 1/(2\Delta t)$. To describe a sine wave two samples per period are necessary. One sample point to define the positive peak and one for the negative peak.

Shannons sampling theorem states that a sampled continuous bandwidth limited function $h(t)$ is completely determined by its samples (measured at $k\Delta t$) if all its frequencies are smaller in magnitude than f_c. More accurately, if $H(f) = 0$ for all $|f| > f_c$, then $h(t)$ is given explicitly by the formula

$$h(t) = \Delta t \sum_{k=-\infty}^{+\infty} h(k\Delta t) \frac{\sin\left(2\pi f_c \left(t - k\Delta t\right)\right)}{\pi(t - k\Delta t)}. \tag{3.3}$$

This is remarkable since the information of a signal with a potentially infinite amount of information is reduced to a finite amount. On the other hand physical signals, such

as audio signals are usually not bandwidth limited. Sampling and applying a discrete Fourier transformation to audio signals will fold the energy contained in the frequencies beyond f_c into the range within f_c. More accurately the energy of the frequency $f_c+\Delta f$ is moved to $f_c-\Delta f$. This is know as *aliasing*. The only way to avoid this is by low-pass filtering the original signal before sampling.

Putting it all together, when down-sampling the 44kHz PCM signal to 11kHz it is necessary to apply a low-pass filter that cuts off frequencies beyond 5.5kHz. From the PCM signal sampled at 11kHz it is possible to obtain the true Fourier transform for frequencies up to 5.5kHz.

The discrete Fourier transform is applied to $h(k\Delta t)$ with $k = 1, \ldots, N$ assuming that the period length of $h(t) = N\Delta t$. For simplicity N is assumed to be even. Since the Fourier transformation is linear, N inputs can only produce N independent outputs, which correspond to the frequencies given by

$$f(n) \equiv \frac{n}{N\Delta t}, \quad n = -\frac{N}{2}, \ldots, \frac{N}{2}. \tag{3.4}$$

Notice that there are N+1 values, however, the two extremes of n are equal. For real valued signals $H(f(n))$ is symmetric resulting in a frequency resolution of

$$f_{res}(n, N, \Delta t) \equiv \frac{n}{N\Delta t}, \quad n = 0, \ldots, \frac{N}{2}. \tag{3.5}$$

The estimation of the discrete Fourier transform which maps N complex numbers to N complex numbers is given by

$$H\left(f\left(n\right)\right) \approx \Delta t \sum_{k=1}^{N} h(k\Delta t)\, e^{2\pi i(k-1)n/N} \tag{3.6}$$

The *Fast Fourier Transformation* (FFT) calculates the discrete Fourier transform in $O(Nlog_2N)$ instead of $O(N^2)$ operations. Especially if N is a power of 2, the Fourier transform can be calculated very efficiently. The Matlab® FFT functions used for this thesis are based on the FFTW[1] library which has been developed at the MIT.

As mentioned above, the assumption is that the function $h(k\Delta t)$ has a period of N. This is rather unlikely when an arbitrary sequence is taken out of a music signal. For example, the signal might start with a high amplitude at $k = 1$ and have a low amplitude at $k = N$. This leads to a discontinuity which gives rise to ringing (leakage) or sidelobes, which can be seen as errors in the frequency-domain. To reduce these effects, window functions $w(k)$ are used and multiplied with the signal function $h(k\Delta t)$ in the time-domain. A common choice for a windowing function is the *Hanning* window. Figure 3.3 depicts the Hanning window and its effect on a signal and its power spectrum. The Hanning window is calculated as

$$w(k) = \frac{1}{2}\left(1 - \cos\left(2\pi\frac{k-1}{N-1}\right)\right), \quad k = 1, \ldots, N. \tag{3.7}$$

[1] http://www.fftw.org

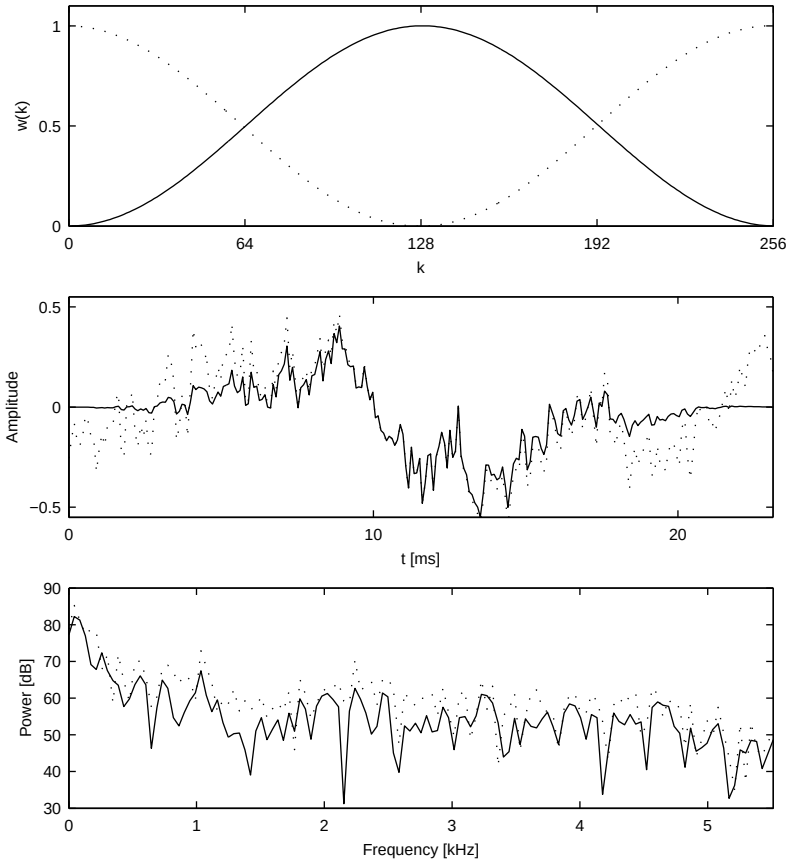


Figure 3.3: The *upper subplot* shows the Hanning function over an interval of 256 samples plotted with the solid line. The dotted lines indicate the values for the Hanning functions with 50% overlap. The *center subplot* depicts the effects of the Hanning function applied to a signal. The signal is the same as the one in Figure 3.2. It is taken from *Freak on a Leash*, sampled at 11kHz, in the interval 60s to 60s+23ms. The solid line is the result of multiplying the Hanning window with the sample values, the dotted line is the original waveform. Notice that while the original signal starts with an amplitude of about -0.25 and ends with an amplitude of 0.17, the signal multiplied with the Hanning function starts and ends at zero. The *lower subplot* shows the effects of the Hanning function in the frequency domain. Again the solid line represents the signal to which the Hanning function has been applied, and the dotted line represents the original signal. The decibel (dB) values will be explained in subsection 3.2.4.

Application

One of the aspects of music is that the frequencies change continuously, however, within very short time frames the frequencies are approximately constant. These very short sequences can be seen as fundamental building blocks of music. Thus, a piece of music can be described with subsequent frequency patterns, each representing a time quantum.

A common choice for this interval is 20ms. The music data used for this thesis is sampled at 11025Hz. To optimize the FFT the number of samples N should be a power of 2. However, this does not mean that it is necessary to have N samples. Shorter signals can be padded with zeros. Using 23ms time frames corresponds to 256 samples and results in a frequency resolution of about 43Hz in a range from 0 to 5.5kHz. As

indicated in Figure 3.3 a 50% overlap between the windows is used. Notice that the sum of the two overlapping Hanning windows at any point always equals 1. A 50% overlap increases the time resolution by a factor 2 to about 12ms. The results of the FFT are visualized in Figure 3.7 as the one-sided power spectral density $P_h(f)$.

The calculations are implemented as follows. The raw audio data is given in vectors $\mathbf{y}_t$ of length N corresponding to 23ms at the time frame t. The power spectrum matrix $\mathbf{P}(n,t)$, where n is the index for the frequency and t for the time frame can be calculated using,

$$\begin{aligned} \mathbf{y}'_t &= \mathbf{W}_N \, \mathbf{y}_t, \\ \mathbf{Y}_t &= \mathrm{fft}(\mathbf{y}_t), \quad \text{and} \\ \mathbf{P}(n,t) &= |\mathbf{Y}_t(n)|^2 \frac{1}{N}. \end{aligned} \tag{3.8}$$

The index n ranges from 1 to $N/2+1$. The matrix $\mathbf{W}_N$ contains the Hanning function weights for N points on the diagonal with zeros elsewhere where $N = 256$ at 11kHz. The *fft* function is taken from the previously mentioned FFTW library.

The data after this feature extraction step basically still has the same size. While the discrete Fourier transformation yields 129 values for 256 sample values, the 50% overlap increases the amount of data by 2.

3.2.2. Critical-Bands

So far a piece of music is represented by a frequency snapshot every 12ms. These have one value every 43Hz starting at 0Hz up to 5.5kHz, where each value represents the power of the respective frequency.

As stated previously, the inner ear separates the frequencies, transfers, and concentrates them at certain locations along the basilar membrane. The inner ear can be regarded as a complex system of a series of band-pass filters with an asymmetrical shape of frequency response. The center frequencies of these band-pass filters are closely related to the critical-band rates. Where these bands should be centered or how wide they should be, has been analyzed throughout several psychoacoustic experiments [ZF99]. While we can distinguish low frequencies of up to about 500Hz well, our ability decreases above 500Hz with approximately a factor of $0.2f$, where f is the frequency. This is shown in experiments using a loud tone to mask a more quiet one. At high frequencies these two tones need to be rather far apart regarding their frequencies, while at lower frequencies the quiet tone will still be noticeable at smaller distances. In addition to these masking effects the critical-bandwidth is also very closely related to just noticeable frequency variations. Within a critical-band it is difficult to notice any variations. This can be tested by presenting two tones to a listener and asking which of the two has a higher or lower frequency.

Since the critical-band scale has been used very frequently, it has been assigned a unit, the *bark*. The name has been chosen in memory of Barkhausen, a scientist who

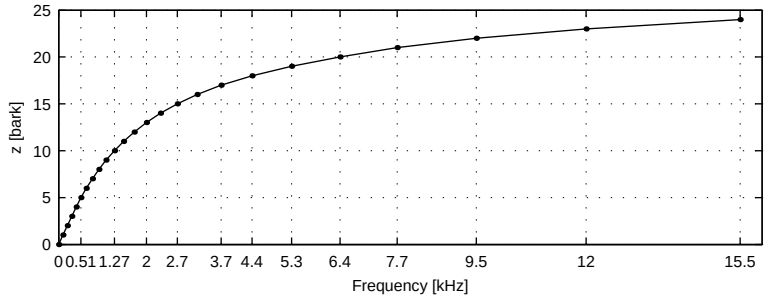


Figure 3.4: The basic characteristics of the critical-band rate scale. Two adjoining markers on the plotted line indicate the upper and lower frequency borders for a critical-band. For example, the 24th band starts at 12kHz and ends at 15.5kHz.

introduced the phon to describe loudness levels for which critical-bands play an important role. Figure 3.4 shows the main characteristics of this scale. At low frequencies below 500Hz the critical-bands are about 100Hz wide. The width of the critical bands increases rapidly with the frequency. The 24th critical-band has a width of 3500Hz. The 9th critical-band has the center frequency of 1kHz. The critical-band rate is important for understanding many characteristics of the human ear.

A critical-band value is calculated by summing up the values of the power spectrum within the respective lower $f_a(i)$ and upper $f_b(i)$ frequency limits of the i-th critical-band. This can be formulated as

$$\mathbf{B}(i,t) = \sum_{n \epsilon I(i)} \mathbf{P}(n,t), \quad I(i) = \{n \mid f_a(i) < f_{res}(n-1, 256, 1/11025) \leq f_b(i)\} \tag{3.9}$$

where i, t, n are indexes and $\mathbf{B}$ is a matrix containing the power within the i-th critical-band at a specific time interval t. $\mathbf{P}$ is the matrix representing the power per frequency and time interval t obtained from Equation 3.8. The relation between actual frequency and the row index n of $\mathbf{P}$ can be obtained using $f_{res}(n-1, 256, 1/11025)$ (cf. Equation 3.5). The necessary upper and lower limits for each critical-band are listed in Table 3.1 [ZF99]. Notice that $\mathbf{P}(1,t)$, which represents the power at 0Hz, is not used.

While the critical-band rate is defined having 24 bands, only the first 20 are used in this thesis, since the highest frequencies in the data are limited to 5.5kHz. The 129 power spectrum values are now represented by 20 critical-bands values. This corresponds to a data reduction by a factor of about 6.5.

3.2.3. Masking

As mentioned before, the critical-bands are closely related to masking effects. Masking is the occlusion of one sound by another sound. A loud sound might mask a simultaneous sound (simultaneous masking), or a sound closely following (post-masking) or preceding (pre-masking) it. Pre-masking is usually neglected since it can only be measured during about 20ms. Post-masking, on the other hand can last longer than 100ms and ends after about a 200ms delay. Simultaneous masking occurs when the test sound and the masker

Table 3.1: Critical-band rate z, lower (f_a) and upper (f_b) frequency limits of the critical bandwidths, f_Δ, centered at f_c [ZF99].

z	f_a, f_b	f_c	z	f_Δ	z	f_a, f_b	f_c	z	f_Δ
Bark	Hz	Hz	Bark	Hz	Bark	Hz	Hz	Bark	Hz
0	0				12	1720			
		50	0.5	100			1850	12.5	280
1	100				13	2000			
		150	1.5	100			2150	13.5	320
2	200				14	2320			
		250	2.5	100			2500	14.5	380
3	300				15	2700			
		350	3.5	100			2900	15.5	450
4	400				16	3150			
		570	4.5	110			3400	16.5	550
5	510				17	3700			
		570	5.5	120			4000	17.5	700
6	630				18	4400			
		700	6.5	140			4800	18.5	900
7	770				19	5300			
		840	7.5	150			5800	19.5	1100
8	920				20	6400			
		1000	8.5	160			7000	20.5	1300
9	1080				21	7700			
		1170	9.5	190			8500	21.5	1800
10	1270				22	9500			
		1370	10.5	210			10500	22.5	2500
11	1480				23	12000			
		1600	11.5	240			13500	23.5	3500
12	1720				24	15500			
		1850	12.5	280					

are present simultaneously. For this thesis the spreading function is used to estimate the effects of simultaneous masking across the critical-bands [SAH79]. The spreading function defines the influence of the j-th critical-band on the i-th and is calculated as

$$\mathbf{S}(i,j) = 15.81 + 7.5(i - j + 0.474) - 17.5\sqrt{1 + (i - j + 0.474)^2}. \tag{3.10}$$

The spread critical-band rate spectrum matrix $\mathbf{B}_S$ is obtained by multiplying $\mathbf{B}$ with $\mathbf{S}$ as follows

$$\begin{aligned} \mathbf{B}_S(i,t) &= \sum_{j=1}^{20} \mathbf{S}(i,j)\,\mathbf{B}(j,t), \quad \text{which is equivalent to} \\ \mathbf{B}_S &= \mathbf{SB}. \end{aligned} \tag{3.11}$$

The simultaneous masking asymmetrically spreads the power spectrum over the critical-bands. The masking influence of a critical-band is higher on bands above it than on those below it.

3.2.4. Decibel

Before calculating sone values it is necessary to transform the data into decibel. The intensity unit of physical audio signals is sound pressure and is measured in *Pascal* (Pa). The values of the PCM data correspond to the sound pressure. It is very common to transform the sound pressure into *decibel* (dB). Decibel is the logarithm, to the base of 10, of the ratio between two amounts of power. The decibel value of a sound is calculated as the ratio between its pressure and the pressure of the hearing threshold given by 20μPa. The sound pressure level in dB is calculated as

$$s_{dB} = 10 \log_{10} \frac{p}{p_0}, \tag{3.12}$$

where p is the power of the sound pressure, and p_0 is the power of the sound pressure of the hearing threshold. The power is calculated as the squared sound pressure.

Parseval's theorem states that the power of the signal is the same whether calculated in the time domain or the frequency domain, so the dB values can be calculated for the spread critical-band matrix $\mathbf{B}_S$. A parameter to adjust is the reference value p_0. Note that the influence of the hearing threshold p_0 on the decibel calculations is non-linear. If the hearing threshold is too low insignificant sounds will become significant, on the other hand if it is too high significant sounds will become insignificant.

Knowing that the sound pressure of the signals is digitized using 16 bit, it could be assumed that the most quiet, just noticeable power of the sound pressure level corresponds to 1 (or -1). When using $p_0 = 1$ the notation db(SPL) is used, where SPL stands for sound pressure level. The maximal decibel value for the energy at a certain frequency using db(SPL) is 96dB. However, the use of this assumption has led to sone values beyond the limit of damage risk in the experiments conducted for this thesis. This occurs because the energy of the frequencies are added together in the critical-bands and the masking function used is additive. The problem with decibel values beyond the limit of damage risk is that only equal loudness contours for levels below the limit are available, thus it is not possible to calculate accurate phon values (see next section), which have a non-linear correlation to the decibel values. To be able to calculate the phon values the PCM amplitudes of the music collection were scaled so that all sounds are below the limit of damage risk. This corresponds to turning down the volume to a level at which the loudness of all music listened to, is within healthy ranges. The hearing threshold parameter p_0 was set to 1/0.35. The loudness matrix in decibel, $\mathbf{L}_{dB}$, is calculated as follows,

$$\begin{aligned} \mathbf{B}'_S(i,t) &= \min(\mathbf{B}_S(i,t), p_0), \quad \text{and} \\ \mathbf{L}_{dB}(i,t) &= 10 \log_{10} \frac{1}{p_0} \mathbf{B}'_S(i,t). \end{aligned} \tag{3.13}$$

Note that the first step is made in order to avoid the logarithm of zero.

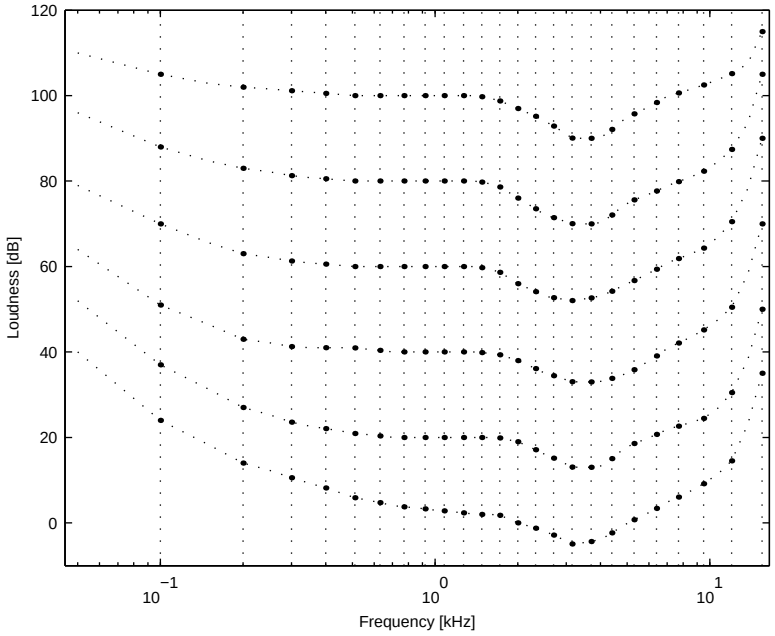


Figure 3.5: Equal loudness contours for 3, 20, 40, 60, 80 and 100 phon. The respective sone values are 0, 0.15, 1, 4, 16 and 64 sone. The dotted vertical lines indicate the positions of the center frequencies of the critical-bands (see Table 3.1). Notice how the critical-bands are almost evenly spaced on the log-frequency axis around 500Hz to 6kHz. The dots correspond to the values of the $\mathbf{C}_{elc}$ matrix used to estimate the phon values from the dB values. The dip around 2kHz to 5kHz corresponds to the frequency spectrum we are most sensitive to.

3.2.5. Phon

The relationship between the sound pressure level in decibel and our hearing sensation measured in sone is not linear. The perceived loudness depends on the frequency of the tone. Figure 3.5 shows so-called loudness levels for pure tones, which are measured in *phon*. The phon is defined using the 1kHz tone and the decibel scale. For example, a pure tone at any frequency with 40 phon is as loud as a pure tone with 40dB at 1kHz. We are most sensitive to frequencies around 2kHz to 5kHz. The hearing threshold rapidly rises around the lower and upper frequency limits, which are respectively about 20Hz and 16kHz.

Although the equal loudness contours are obtained from experiments with pure tones, they are frequently applied to calculate the specific loudness of the critical-band rate spectrum. The loudness matrix in phon, $\mathbf{L}_{phon}$, can be calculated using the equal loudness contour matrix $\mathbf{C}_{elc}$ and the corresponding phone values to each contour $\mathbf{c}_{phon} = [3, 20, 40, 60, 80, 100]$. $\mathbf{C}_{elc}(i, j)$ contains the decibel values of the j-th loudness contour at the i-th critical-band. The values are illustrated in Figure 3.5 and can be found in Appendix A.2.1. Values in between two equal loudness contours are

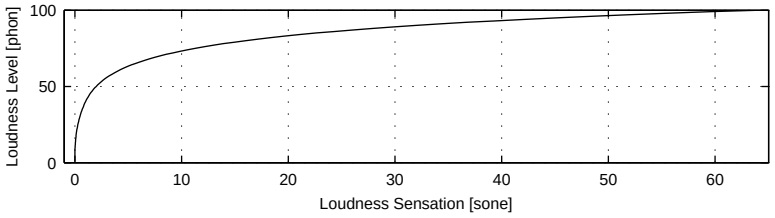


Figure 3.6: The relationship between the loudness level and the loudness sensation.

interpolated linearly, as follows

$$\begin{aligned}
\mathbf{L}'_{dB}(i,t) &= \max(\mathbf{L}_{dB}(i,t), \mathbf{C}_{elc}(i,1)),\\
level_{i,t} &= \operatorname*{argmin}_{j}(\mathbf{L}'_{dB}(i,t) < \mathbf{C}_{elc}(i,j)),\\
r_{i,t} &= \frac{\mathbf{L}'_{dB}(i,t) - \mathbf{C}_{elc}(i, level_{i,t}-1)}{\mathbf{C}_{elc}(i, level_{i,t}) - \mathbf{C}_{elc}(i, level_{i,t}-1)}, \quad \text{and}\\
\mathbf{L}_{phon}(i,t) &= \mathbf{c}_{phon}(level_{i,t}-1) + r_{i,t}\,\mathbf{c}_{phon}(level_{i,t}).
\end{aligned} \tag{3.14}$$

Note that $\mathbf{L}_{phon}(i,t)$ can only be calculated if there exists an equal loudness contour j in $\mathbf{C}_{elc}(i,j)$ so that $\mathbf{c}_{phon}(j) > \mathbf{L}_{phon}(i,t)$. The highest decibel level used here is 100dB and p_0 (see above) is adjusted manually so all sounds are below this limit.

3.2.6. Sone

Finally, from the loudness level $\mathbf{L}_{phon}$ the specific loudness sensation $\mathbf{L}_{sone}$ per critical-band, following [Bla81], is calculated as,

$$\mathbf{L}_{sone}(i,t) = \begin{cases} 2^{\frac{1}{10}\left(\mathbf{L}_{phon}(i,t)-40\right)} & \text{if } \mathbf{L}_{phon}(i,t) > 40\\ \left(\frac{1}{40}\mathbf{L}_{phon}(i,t)\right)^{2.642} & \text{otherwise.} \end{cases} \tag{3.15}$$

The relationship between phon and sone can be seen in Figure 3.6. For low values up to 40 phon the sensation rises slowly until it reaches 1 sone at 40 phon. Beyond 40 phon the sensation increases at a faster rate. The highest values that occurred in the experiments conducted for this thesis are below 60 sone, due to the adjustment of the threshold in quiet p_0.

Figure 3.7 demonstrates the effects of the different loudness calculations using the previously introduced titles *Für Elise* by *Beethoven*, and *Freak on a Leash* by *Korn*. Notice that the maximum specific loudness of *Freak on a Leash* is about 4 times higher then the maximum of *Für Elise*. Furthermore, *Freak on a Leash* covers the whole critical-band rate spectrum and seems to create a chaotic sensation pattern while *Für Elise* is primarily active only in the critical-bands from 3 to 9 bark.

3.2.7. Examples

The methods applied so far have mainly been developed based on experiments with very simple sounds. Alternative ways to calculate $\mathbf{L}_{sone}$ include models which simulate each

part of the ear, from the deflections of the outer ear to the activations of the neurons in the auditory nerve [DP96]. A Matlab® toolbox, namely HUTear[2] [HP99], which implements such models has been developed at the Helsinki University of Technology.

On the other hand, the methods described in the previous sections have successfully been applied several times, for example, in [FG94, Yan99]. Similar methods are frequently used especially in the speech processing community [GM99]. However, this is not a guarantee that these methods work in this context. It is important to fully understand what this series of transformations has done with the data, and what exactly the data "looks like" before the data-mining process is continued and further feature extraction steps are applied.

To obtain a better understanding of the preprocessing steps so far, they are illustrated in Figures 3.7 and 3.8. On the left side of Figure 3.7 is the first 6-second sequence extracted from *Beethoven, Für Elise*. Starting at about the 2nd second, the main theme can be seen. It is a quiet piano piece, only one tone at a time is played. On the right side is the first 6-second sequence extracted from *Freak on a Leash* by *Korn*. A distorted voice articulates strange sounds while instruments including drums and probably at least one electric guitar play a scratchy theme, all in all a very interesting and unusual sound experience.

PCM Audio Signal

The first row in Figure 3.7 represents the PCM data introduced in Section 3.1. Although the amplitude cannot be directly expressed as a physical unit, the amplitude values correspond to sound pressure values. While *Für Elise* hardly reaches 5% of the maximum amplitude at its highest sound pressure intensities, the amplitude of *Freak on a Leash* comes close to the maximum several times. This is also reflected in the subsequent plots and can be seen from the color-bars to the right of each plot. The intensities reach a maximum of about 60dB or 6 sone on the left, and about 80dB or 25 sone on the right side. Note that the original difference in amplitude by a factor of 20 only causes a sensation difference of a factor of about 4. As part of the experiments conducted for this thesis the PCM data was originally scaled so that the maximum value of a piece of music equals the maximum possible amplitude. This was later dismissed since it showed no significant positive effects on the final results. Intuitively, it can be argued that the volume settings of the speakers are usually not changed while listening to different pieces of music. If there are significant differences in the sound intensities, then these might be important characteristics of the respective pieces of music.

Power Spectrum

The power spectrum subplots in Figure 3.7 represent 511 time frames each with 129 frequency values. Each time frame represents about 23ms and overlaps half of the

[2] `http://www.acoustics.hut.fi/software/HUTear`

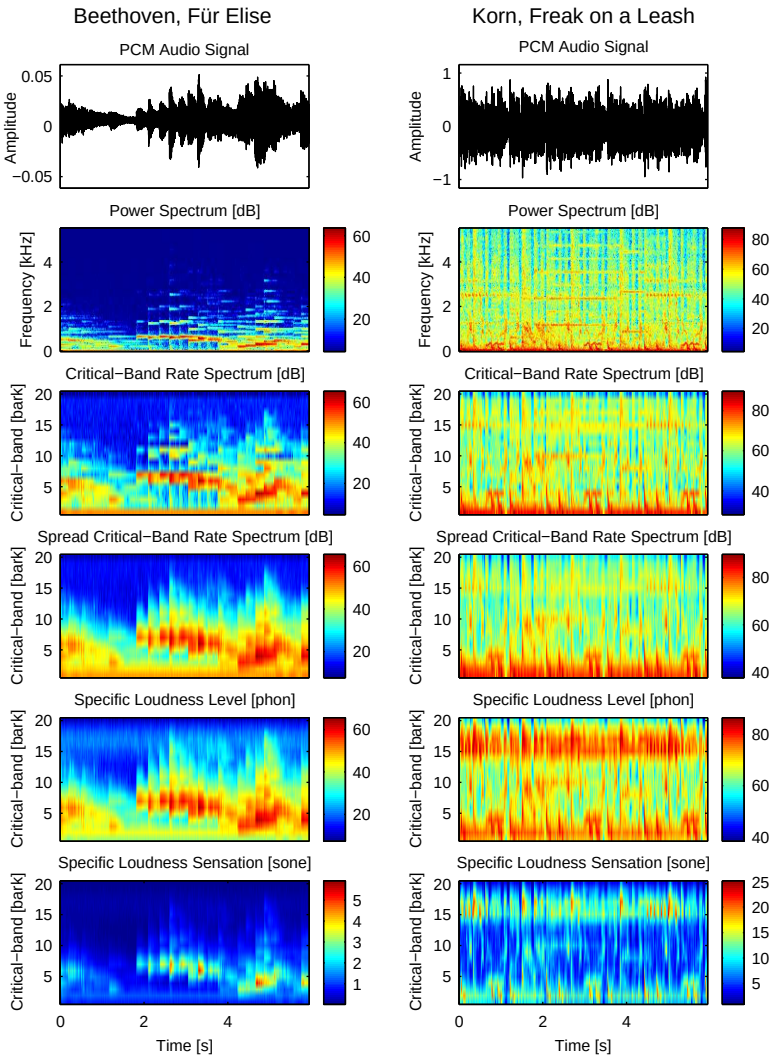


Figure 3.7: The feature extraction steps from the 11kHz PCM audio signal to the specific loudness per critical-band.

preceding and half of the succeeding time frame. The pure power spectrum would not reveal much information, which is why the decibel values are used. The values displayed $\mathbf{P}'$ have the following relationship to $\mathbf{P}$,

$$\mathbf{P}'(i,t) = 10 \log_{10} \frac{1}{p_0} \min(p_0, \mathbf{P}(i,t)). \tag{3.16}$$

On the left side the harmonics are very clearly visible as the equidistant, parallel, horizontal lines, which become weaker at higher frequencies. With a higher frequency resolution, for example, by increasing the sampling frequency or using a wider time frame, it would be possible to determine the exact pitch of the piano tones played (e.g. [Ell96]). However, the exact pitch is not relevant for the genre of a piece of music.

The harmonics of *Für Elise* can also be used to estimate the intensities of the tones played. The further up on the frequency scale the harmonics are still visible, the more intense the keynote has been played.

The power spectrum of *Freak on a Leash* suggests that there is not much variation

in the music sequence. The main characteristic seems to be a texture of vertical lines, indicating the beats. However, there are also some harmonics visible as vertical lines. Very typical is how the whole spectrum is active and that the power intensities are highest in the lower frequencies.

As mentioned previously in this chapter, the audio data is down-sampled to 11kHz resulting in a Nyquist frequency of about 5.5 kHz. The main active frequencies of *Für Elise* are contained within this spectrum which corresponds to what can be seen in the subplot. On the other side *Freak on a Leash* contains high frequencies that are not within the spectrum, as the subplot suggests the texture continues over the entire audible spectrum. However, simple tests using average PC speakers revealed no difference in the sound sensation of *Freak on a Leash* at 11kHz or 44kHz sampling rate. Notice how the average specific loudness sensation decreases from bark 17 to bark 20. Thus, the limit of 5.5kHz is suitable for *Freak on a Leash*.

Critical-Band Rate Spectrum

These subplots of Figure 3.7 contain about 85% less values than the power spectrum plots, and yet they preserve the main characteristics. Especially the temporal resolution is the same, which is important since the determination of the dynamics of a piece of music later on relies on high temporal resolutions.

The values of the critical-band rate spectrum $\mathbf{B}$ have been transformed to decibel just as those of the power spectrum, and all subsequent are, expect for the specific loudness sensation. Notice that the decibel range slightly increases as the frequencies are bundled together into critical-bands.

Generally it appears that the data has become fuzzy. The harmonics of *Für Elise* are hardly recognizable while some diffuse activities have become more apparent. On the other hand the keynotes of *Für Elise* are still clearly recognizable. Notice that a deficiency has become more apparent as well. When listening to the sequence there is no bass noticeable, however, the subplot shows a significant activity at bark 1, it is present in the power spectrum, but being limited to the lowest frequencies it is hardly visible. This seems to be some error in the recording but does not influence the perceived sound quality. Notice also how this deficiency gradually disappears in the subsequent processing steps.

The critical-band rate spectrum of *Freak on a Leash* reveals an effect caused by low-pass filtering the music up to 5.5Hz. While throughout bark 1 to 19 the intensities are rather continuous, at bark 20 there is a discontinuity. The values of the 20th critical-band are clearly lower compared to the 19th. The answer to this can be found looking at Table 3.1. The 20th critical-band is defined from 5.3kHz to 6.4kHz. Despite this apparent discontinuity the information necessary to determine the genre is still contained in the data, in the same way, as the genre can be determined listening to the low-pass filtered music.

Spread Critical-Band Rate Spectrum

These subplots of Figure 3.7 have the same frequency resolution as the previous, however, a spreading function is applied which smoothens the critical-band intensities. Notice that the spreading function [SAH79] leads to slightly higher decibel values.

The spreading function is asymmetric with a stronger influence on the bands above than on those below the one it is centered on. It is also important that the spreading function does not influence the temporal resolution. Post-masking effects, which smoothen the temporal dimension, have been ignored since they showed negative influences on the determination of the exact beat intervals.

Looking at the spread critical-band rate spectrum it is not possible to recognize the harmonics of the tones. While the harmonics influence our sensation an untrained listener is not able to identify them exactly. Instead, an untrained listener might perceive the harmonics as shadows of the keynote that add to tones richness.

Wonho Yang [Yan99] mentions that spreading the critical-bands might not have a positive impact when trying to measure the objective speech quality. Generally some perhaps very typical and important features might be lost through the smoothing. However, the context of this thesis is limited to the dynamics of a piece of music. In this context the spreading is useful, since it is not a significant difference if a piece of music is dynamic at bark 10 or at bark 11.

Specific Loudness Level - Phon

The specific loudness level subplots in Figure 3.7 reveal the effects of the equal loudness level transformation for the critical-bands. While the intensities in the lower frequencies are reduced the intensities around bark 18 are increased. *Für Elise* does not use the frequencies around 4kHz (which corresponds to the 18th critical-band). The constant activity in the background, which becomes apparent, is another deficiency in the recording. This deficiency seems to be caused by the generally very quiet recording, which enables broadband noise to become noticeable, especially in the frequency spectrum we are most sensible too, which is around 4kHz. However, in the same way as this deficiency does not appear to be significant in the illustrated specific loudness level images, it does not influence the perceived genre when listening to the recording of *Für Elise*.

On the other side *Freak on a Leash* uses the frequencies around 4kHz frequently. Some not recognizable instruments and some parts of the distorted male voice are active in that area.

Notice how the magnitude of the decibel values has not changed significantly throughout the transformation steps applied so far.

Specific Loudness Sensation - Sone

The final results of the loudness calculation can be seen in the last row of subplots in Figure 3.7. The image of *Für Elise* has become clear again and the single tones can be

recognized. The data corresponds quite well to the perceived music. The first 2 seconds of the sequence are rather quiet, three quiet tones are played right at the start and then there is a short pause. The first is the most intense at about bark 5. The second one follows shortly after and is played very faint. It can be seen around bark 4 to 5 with a very short pause to the first tone. The third tone is played a little stronger than the second and can be seen around bark 3. Then there is the pause that is ended as the main theme of *Für Elise* starts to play. Notice the quickly played gentile variations in the critical-band spectrum around bark 6 to 7 before the tones are played slower and drop down to bark 5. Then in three steps, starting at bark 3, the tones slowly rise back to bark 5.

The sound of *Freak on a Leash* is far more complex and cannot be analyzed in a similar way. Although there is something like a melody, which is played throughout the sequence, it is hard to localize it in the specific loudness sensation plot. The melody is not the main characteristic of this sequence, whereas the strange sounding soundscape definitely is. Something that can be noticed is the recurrent pattern at the low frequencies. Three times there is a larger pause with a specific pattern right before it. Between these there are two smaller pauses. These are followed immediately by a sound running down the frequencies from about bark 5 to 2. And these again, are followed every time in the same distance, by a sound running down the frequencies from bark 5 to 2, although not as intense.

Further Comparison

In the previous subsections the differences between the two pieces of music compared to each other were obvious. However, not all music is as clearly distinguishable. Figure 3.8 depicts the feature extraction steps applied to the famous song *Yesterday* from the *Beatles*, and *Rock DJ* from *Robbie Williams*, which has been in the charts recently. As in Figure 3.7 the first 6second segment is used. Within these 6 seconds *Robbie Williams* is singing: *"... getting high and the girls even more so. Wave your hands if you're not with a man. Can I kick it? ..."*. His voice is accompanied by a heavy, but not fast beat with a very low pitch. To each bass beat there is a response which sounds somehow like a plucked chord with a higher pitch than the bass. And a piano plays a simple melody on top of some of the beats. The 6-second sequence of *Yesterday* contains: *"... now it looks as though there here to stay ..."*, starting a little after the first second of the sequence and ending before the fifth second. This part is sung slowly by a male voice and is accompanied only by a guitar playing a few chords.

The PCM subplot of *Yesterday* seems to have similarities with *Rock DJ* and *Für Elise*. While the PCM subplot of *Rock DJ* seems to be between *Yesterday* and *Freak on a Leash*. Unlike *Für Elise* both have amplitude values covering the whole spectrum, *Rock DJ* a little more so than *Yesterday*.

The power spectrum of *Rock DJ* reveals the importance of the beat for the respective music sequence. While the power spectrum of *Yesterday* shows countless harmonics in

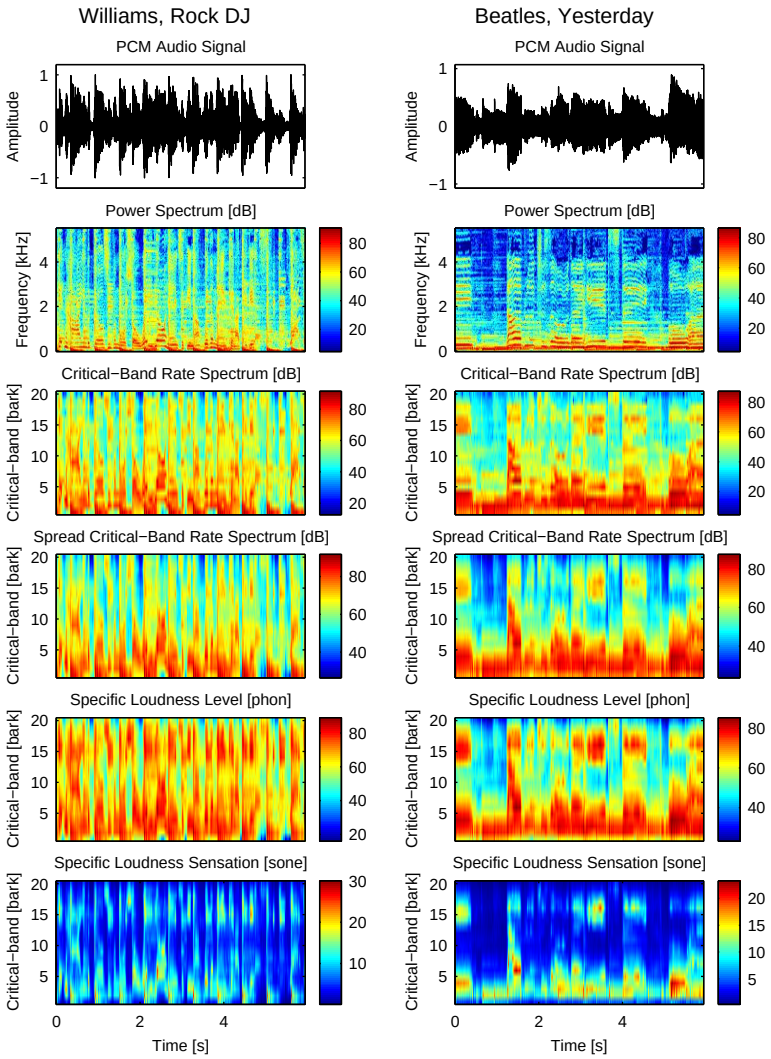


Figure 3.8: Two pieces of music that are not as different as the two in Figure 3.7.

the range of 2kHz to 4kHz, created by the singer. For example the harmonics after the 4th second correspond the sung word "stay", just before it, the word "to" is sung rather quite. And before that the word "here" is sung, its harmonics are clearly visible as well. The exact pattern in the range of 2kHz to 4kHz is very interesting for speech recognition, however, for assigning a piece of music to a genre it is not relevant.

This is underlined by the subsequent plots. The patterns created by the harmonics are not recognizable any longer. Looking at the specific loudness sensation it is only possible to determine when a word has been sung and at about which frequency. For example, the first word *"now"* (the very first visible word type sound event is a remaining of *"away"* but only an *"e"* is hearable), has a significant variation in its frequency. Starting at about 16 bark it drops down to 6 bark.

Comparing all four 6-second sequences in the Figures 3.7 and 3.8 the following can be said in regard to their dynamics. *Beethoven's Für Elise* is faintly active in the frequencies around bark 3 to 9. It uses a lot of variations in the speed, there is hardly any recurrent beat pattern. *Freak on a Leash* has a beat pattern, however, most of the

activities seem rather random, and are confined to the spectrum around bark 16 and around bark 2. *Rock DJ*'s activities are spread out a little more evenly over almost the whole spectrum. *Rock DJ* also has a very strong, but not as fast beat. *Yesterday* has a constant activity around and below bark 5. Although there does not seem to be a strong beat pattern such as in *Rock DJ*, or *Freak on a Leash*, the beats are far stronger than in *Für Elise*.

3.3. Dynamics

So far each song is represented by several 6-second sequences of the specific loudness sensation per critical-band, $\mathbf{L}_{sone}(i,t)$. It would be possible to use $\mathbf{L}_{sone}(i,t)$ to calculate similarities between the data. One option would be to compare two sequences $\mathbf{L}^a_{sone}$ and $\mathbf{L}^b_{sone}$ point-wise, i.e. comparing $\mathbf{L}^a_{sone}(i,t)$ and $\mathbf{L}^b_{sone}(i,t)$ for all i and t. The result might be quite surprising. For example, shifting *Rock DJ* (Figure 3.8) by only 40ms would result in a huge difference to the un-shifted sequences - although they sound the same. The same problem would occur for any of the other sequences presented in the previous section. Thus, the final representation of the data must be invariant to time shifts.

As mentioned before, the aim is to gather information on the dynamics of a sequence. Weil [Wei99] and Frühwirth [Frü01] have used the Fourier transforms of the activities in the frequency bands, which are also used here. Ellis [Ell96] has shown that using a similar concept it is possible to predict the pitch of a sound. Ellis analyzed periodically recurring peeks in the loudness of a frequency band by calculating the autocorrelation. The main difference between Weil, Früwirth and Ellis is the frequency ranges they analyze. While Ellis analyzes patterns with periods of about one millisecond (which corresponds to frequencies up to 1kHz), Frühwirth limits his investigation to frequencies up to 25Hz. Weil uses a similar spectrum as Frühwirth, though limits his analysis to 15Hz.

3.3.1. Amplitude Modulated Loudness

The loudness of a critical-band usually rises and falls several times. Often there is a periodical pattern, also known as the rhythm. At every beat the loudness sensation rises, and the beats are usually very accurately timed.

The loudness values of a critical-band over a certain time period can be regarded as a signal that has been sampled at discrete points in time. The periodical patterns of this signal can then be assumed to originate from a mixture of sinuids. These sinuids modulate the amplitude of the loudness, and can be calculated by a Fourier transform.

An example might illustrate this. To add a strong and deep bass with 120 *beats per minute* (bpm) to a piece of music, a good start would be to set the first critical-band (bark 1) to a constant noise sensation of 10 sone. Then one could modulate the loudness

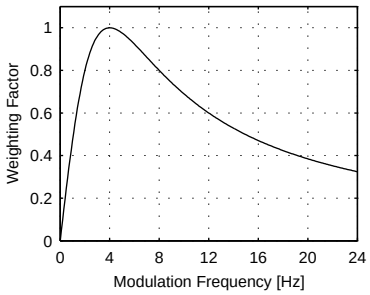


Figure 3.9: The relationship between fluctuation strength and the modulation frequency.

using a sine wave with a period of 2Hz and an amplitude of 10 sone.

The modulation frequencies, which can be analyzed using the 6-second sequences and time quanta of 12ms, are in the range from 0 to 43Hz with an accuracy of 0.17Hz. Notice that a modulation frequency of 43Hz corresponds to almost 2600bpm.

The modulation amplitude $\Delta\mathbf{L}_i(n)$ with the frequency $f_{res}(n, 512, 256/11025/2)$ (cf. Equation 3.5) of the i-th critical-band is calculated as follows,

$$\Delta\mathbf{L}_i = \text{fft}(\mathbf{L}_{sone}(i, 1 \ldots 511)), \tag{3.17}$$

where $\mathbf{L}_{sone}(i, 1 \ldots 511)$ is a 6-second sequence of the the i-th critical-band of any piece of music. The *fft* function is the same as in Equation 3.8. Since there are only 511 values for the FFT, the signal is padded with one zero.

3.3.2. Fluctuation Strength

The amplitude modulation of the loudness has different effects on our sensation depending on the frequency. The sensation of *fluctuation strength* [Ter68, Fas82] is most intense around 4Hz and gradually decreases up to a modulation frequency of 15Hz (cf. Figure 3.9). At 15Hz the sensation of *roughness* starts to increase, reaches its maximum at about 70Hz, and starts to decreases at about 150Hz. Above 150Hz the sensation of hearing *three separately audible tones* increases [ZF99].

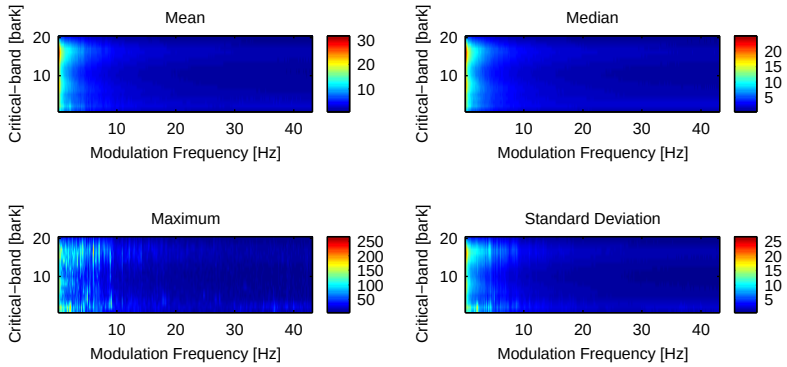
The fluctuation strength of a tone with the loudness ΔL, which is 100% amplitude modulated with the frequency f_{mod} can be expressed by,

$$f_{flux}(\Delta L, f_{mod}) \propto \frac{\Delta L}{(f_{mod}/4\text{Hz}) + (4\text{Hz}/f_{mod})}. \tag{3.18}$$

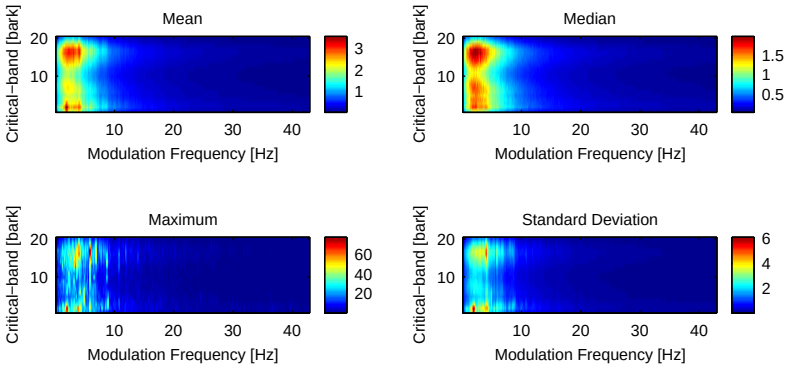
The modulation amplitudes $\mathbf{F}(i, n)$ of the i-th critical-band and the modulation frequency $f_{res}(n, 512, 128/11025)$ (cf. Equation 3.5) are weighted according to the fluctuation strength sensation as follows,

$$\mathbf{F}(i, n) = f_{flux}(|\Delta\mathbf{L}_i(n+1)|, f_{res}(n)), \tag{3.19}$$

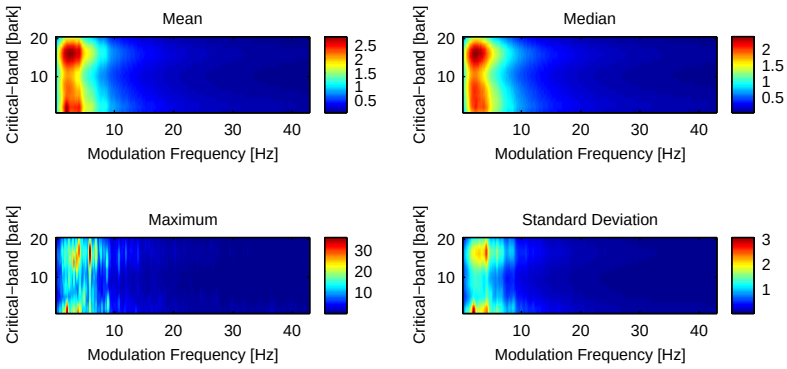
Notice that $\Delta\mathbf{L}(0)$, with $f_{res}(0, 512, 128/11025) = 0$Hz, is ignored since it does not influence the fluctuation.



(a) Modulation Amplitude Spectrum



(b) Fluctuation Strength Spectrum



(c) Modified Fluctuation Strength Spectrum

Figure 3.10: Basic statistics of all 6-second sequences in the music collection used.

Figure 3.10 depicts the effect of calculating the fluctuation strength. The modulation amplitudes (cf. Figure 3.10(a)) are highest at the lowest modulation frequencies. All other frequencies would play a minor role in calculating the distances between two sequences.

The fluctuation strength (cf. Figure 3.10(b)) is spread out more evenly across the spectrum. Notice that the standard deviation shows clear patterns of vertical lines, which are caused by sequences with a strong beat at the corresponding modulation frequencies. Around bark 1 to 3 there are some highlights, these are typical bass

patterns.

As mentioned above the highest modulation frequency which can be calculated is about 43Hz. However, music will usually not show much fluctuation at frequencies beyond 10Hz (600bpm). Figure 3.10(b) shows the standard deviation of the fluctuation strength of all sequences. It rapidly decreases after about 5Hz and there is hardly any fluctuation around 15Hz. The mean, median, and maximum values of the fluctuation strength confirm that there is not much activity beyond 10Hz. To reduce the amount of data only the frequency values up to 10Hz are used, which are the first 60 values.

3.3.3. Modified Fluctuation Strength

The fluctuation strength can be used for point-wise comparison of two music sequences. Each $\mathbf{F}(i, n)$ has a meaning which does not depend on time shifts. For example, the value at $\mathbf{F}(1, 8)$ describes the fluctuation strength at the first critical-band (bark 1) with the modulation frequency of 1Hz, which corresponds to a bass with 60bpm.

The 3-dimensional matrices $\mathbf{F}$ can be treated as images as they are, for example, in Figure 3.11. Individual characteristics of these images can be emphasized or suppressed using image processing methods. A similar approach where image processing techniques are applied to music data was presented by Mellinger [Mel91]. However, unlike in this thesis, Mellinger's focus was on sound source separation in the context of auditory scene analysis and music transcription.

There are two basic types of fluctuation patterns. In Figure 3.11 this can be seen. Classical music does not tend to have one dominant periodical beat pattern. It usually has a lot of variation in its tempo. On the other hand contemporary rock music usually contains at least one strong specific beat. This is reflected in the specific fluctuation strength in such a way that rock music has sharp vertical lines. These often range over all the critical-bands, sometimes only in the lower frequencies, sometimes only in the higher frequencies. On the contrary, for example, *Für Elise* or similar music will have a more spread fluctuation over several frequencies close to each other. To distinguish these two types of patterns more clearly a gradient filter is used to emphasize sharp vertical lines.

After emphasizing vertical lines it is necessary to suppress irrelevant characteristics, in particular the exact location of peeks. This can be illustrated, for example, using three 6-second music sequences given as $\mathbf{F}^a$, $\mathbf{F}^b$, and $\mathbf{F}^c$. For simplicity they shall contain all zeros except for $\mathbf{F}^a(2, 8) = 10$ sone and $\mathbf{F}^b(1, 9) = 10$ sone. The sequences $\mathbf{F}^a$ and $\mathbf{F}^b$ both have a bass beat, with almost the same frequency. However, using for example the Euclidian vector norm, the sequences $\mathbf{F}^a$ and $\mathbf{F}^b$ would be double as far apart from each other as they are from $\mathbf{F}^c$. More intuitive would be if $\mathbf{F}^a$ and $\mathbf{F}^b$ would be much closer to each other than they are to $\mathbf{F}^c$. To be able to use the Euclidean vector norm and recognize such similarities some spreading is applied. In the above example $\mathbf{F}^a$ is smoothened so that, for example, the peek at $\mathbf{F}^a(2, 8)$ is spread first over the critical-bands up and down 4 bands and then over the modulation frequencies with

Gaussian weighting factors. The same is done for the peek at $\mathbf{F}^b(1,9)$ resulting in the desired effects regarding their Euclidean distances.

The modified fluctuation strength (MFS) $\mathbf{F}_{mod}$ is calculated as follows,

$$\begin{aligned} \mathbf{F}'(i,n) &= |\mathbf{F}(i,n) - \mathbf{F}(i,n+1)|, \quad \text{and} \\ \mathbf{F}_{mod} &= \mathbf{S}_i \mathbf{F}' \mathbf{S}_n. \end{aligned} \tag{3.20}$$

The first part of this equation calculates the difference between two adjacent modulation frequencies n and $n+1$, this emphasizes the vertical lines, which correspond to specific beats. The values for the matrixes $\mathbf{S}_i$ and $\mathbf{S}_n$ can be found in Appendix A.2.2. $\mathbf{S}_i$ is a 20 by 20 matrix, defining how the values of $\mathbf{F}'$ are spread over the critical-bands. $\mathbf{S}_n$ is a 60 by 60 matrix, with the spreading values for the modulation frequencies. The index n only runs from 1 to 60, ignoring the modulation frequencies beyond 10Hz.

Figure 3.10(c) shows the basic statistics of the MFS. The main difference is the range of the maximum and standard deviation values. Since high peeks are spread across the neighboring modulation frequencies and critical-bands, their peek values decrease, and so does the standard deviation. The main characteristics, such as the vertical lines, which can be seen in the maximum and standard deviation statistics, are not significantly modified.

3.3.4. Examples

In Figure 3.11 the already introduced sequences of *Korn, Freak on a Leash* and *Beethoven, Für Elise* are used to demonstrate the relationship between the specific loudness given by $\mathbf{L}_{sone}$ and the MFS $\mathbf{F}_{mod}$. Only the first 60 values (excluding the 0Hz value) are used, which corresponds to the frequencies up to 10Hz. Notice the difference in the ranges of the values. The application of the gradient filter reduces the fluctuation strength of *Für Elise*, while the main vertical line a little below 7Hz in *Freak on a Leash* is emphasized.

The effects of calculating the fluctuation strength from the modulation amplitudes can also be seen very clearly in Figure 3.11. Looking only at the modulation amplitude of *Für Elise* is seems as though there is no beat. The rhythmic event plot (details see below) indicates that there are several rhythmic events with the average frequency of about 3.6Hz. In the fluctuation strength subplot the modulation frequencies around 4Hz are emphasized more. There are no clear vertical lines though, since there are no periodic beats recurring with the exact distance in between. On the other side *Freak on a Leash* has a very clear vertical line, corresponding to the sound of drums which add periodical beats to the music.

Model of Rhythm

To compare the results obtained using the fluctuation strength with models based on the overall loudness, the model of rhythm presented in [ZF99] is used.

A rhythmic event is defined as a maximum in the loudness-time function. More specifically the model postulates that only maxima above a value of $0.43N_m$ are used,

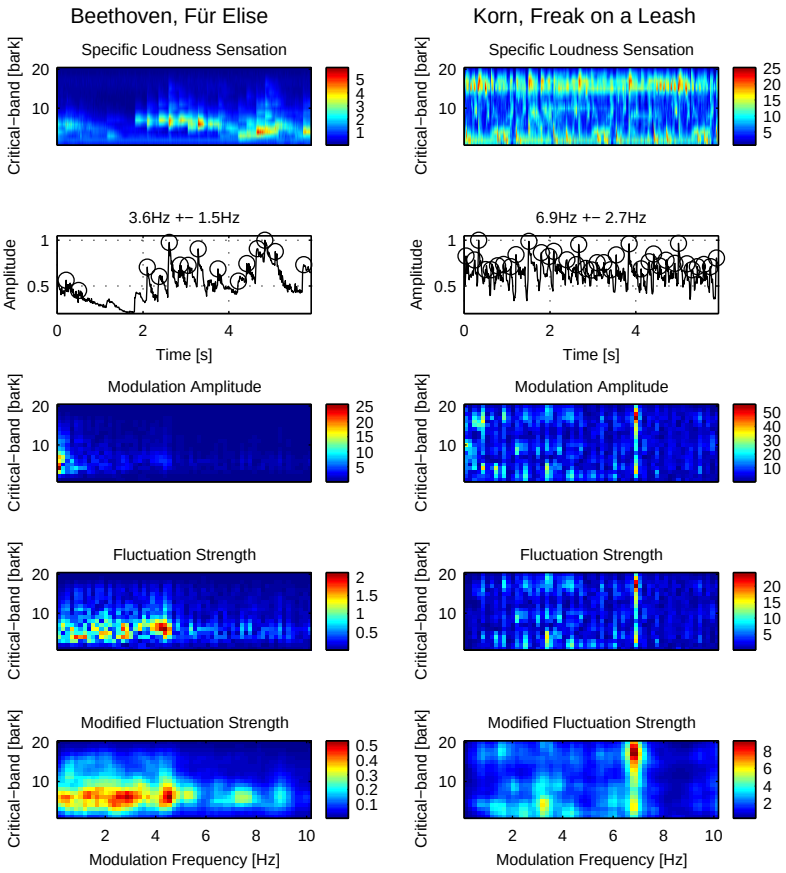


Figure 3.11: Fluctuation strength comparision between the 6-second sequences of *Beethoven, Für Elise* and *Korn, Freak on a Leash*. The second row of subplots represent the rhythmic events. They are titled with the average and the standard deviation of the distances between the rhythmic events. The plotted line corresponds to the sum of the specific loudness in each critical-band. The circles mark the rhythmic events.

where N_m represents the loudness of the highest maximum within a relevant time, which is assumed to be the interval of 6 seconds here.

A second condition is that only relative maxima of sufficient height are considered. If $\Delta N < 0.12 N_m$ then the maxima are ignored, where ΔN is the difference in loudness between the last minimum and the respective maximum.

As a final condition, only maxima which are separated more than 120ms are considered to be separate rhythmic events. If there are several maxima within this time frame, only the highest maxima is selected.

Notice that in Figure 3.11, in the rhythmic events subplot belonging to *Für Elise*, almost all the keynotes are found, except for the one very quietly played right after the first. While on the other side *Freak on a Leash* has so many rhythmic events, that it seems difficult to recognize any pattern. Notice that there are 5 rhythmic events that are louder than all others (very close to the maximum amplitude). These define the main beat.

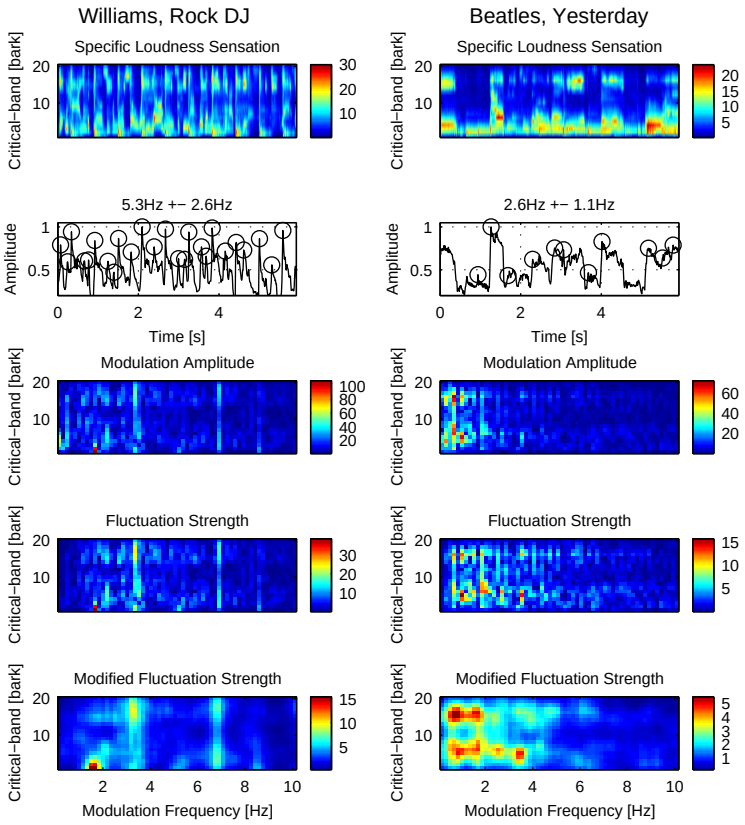


Figure 3.12: Fluctuation strength comparison between the 6-second sequences of *Robbie Williams, Rock DJ* and *Beatles, Yesterday*. The second row of subplots represent the rhythmic events. They are titled with the average and the standard deviation of the distances between the rhythmic events. The plotted line corresponds to the sum of the specific loudness in each critical-band. The circles mark the rhythmic events.

3.4. Computational Efficiency

As mentioned previously one aspect of good features representing music is that they should be computationally efficient as music collections grow large in size rapidly.

In this section some practical issues are summarized related to the feature extraction process. All time statements refer a P2 350MHz with 256MB RAM using Matlab® 6 and compiled Matlab® functions.

The original 359 MP3 files, which are used for the experiments conducted for this thesis, occupy about 1.3GB on the hard disc and represent about 23 hours of music. Winamp[3] was used to convert the MP3 format to the 11kHz mono PCM format. The PCM data is known as WAV in Microsoft® systems and as AU in Unix systems. The PCM data occupies about 1.7GB. The conversion only takes a few minutes.

The PCM data is then read by Matlab® and the results of the intermediate steps are saved to the disk. The 3940 selected 6-second sequences, stored as PCM data, occupy about 490MB of storage. The specific loudness values are stored for each piece

[3] `http://www.winamp.com`

of music separately in the Matlab® MAT format. Together they occupy 150MB of storage. The fluctuation strength data matrix with all 256 modulation frequencies has (stored in single precision) about 80MB. Matlab® takes about 2 hours for a complete feature extraction run.

3.5. Summary

In this chapter stepwise features have been extracted from the raw music data. Each step has been described based on its psychoacoustical background and explained intuitively. The resulting features are robust in regard to variations that do not influence the beat characteristics. Moreover, the methods described are computationally not very expensive.

Figure 3.13 summarizes these steps and shows where and which critical parameters have been chosen. Starting with *music* given in any digital form, the first step is to down-sample the music and combine the two stereo channels into one mono channel, represented in the *PCM format*. Then a discrete Fourier transform is used to obtain the *power spectrum* of the signal. The parameters defining the window function (Hanning) and the length of the time frame (256 samples) are critical but common choices. The frequencies are then bundled into 20 *critical-bands*, where they are *spread* asymmetrically before the power values are transformed into *decibel* values. The reference value for the decibel value is adjusted so that there are no sounds with a loudness beyond the limit of damage risk. The decibel values are then transformed into *phon* values, which in turn are transformed to *sone* values, which represent the specific loudness sensation of a piece of music in the dimensions of 23ms time intervals and critical-bands. Using the discrete Fourier transformation again, the *amplitude modulations* of the loudness patterns in the time domain are calculated. These are the basis for the *loudness fluctuation strength*. To emphasize sharp edges the fluctuation strength is modified using a gradient filter. To be able to apply a Euclidean metric and yet recognize similarities between modulation frequencies and critical-bands, finally a Gaussian filter is applied to obtain the *modified fluctuation strength* (MFS).

At the end of the feature extraction process each piece of music is represented by several 6-second sequences. Each of these sequences is stored as a matrix $\mathbf{F}_{mod}(i, n)$ representing the modified fluctuation strength value for the i-th critical-band and the modulation frequency $f_{res}(n, 512, 128/11025)$ (cf. Equation 3.5). The MFS is time invariant. It ignores not reoccurring elements in the specific loudness sensation and emphasizes elements that reoccur in exactly the same intervals. The MFS includes all information necessary to determine the rhythm of a piece of music, while other information has been filtered out.

It thus offers itself as a basis for comparing the perceived similarity between two 6-second sequences of music. In the next chapter a brief evaluation will be presented regarding the suitability of the MFS to calculate the similarity of sequences. Further-

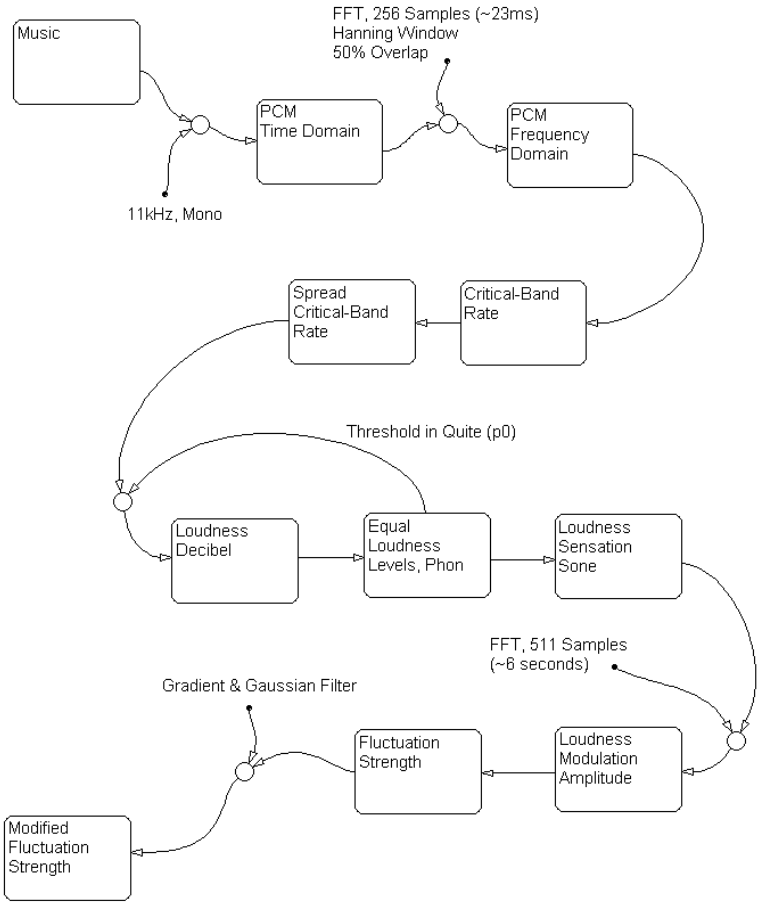


Figure 3.13: Summary of the feature extraction steps and the critical parameters used.

more, the problem of representing one piece of music based on the features extracted from its sequences will be dealt with.

4. Clustering

The previous chapter dealt with the extraction of features from the raw music data. In this chapter the music collection (Appendix B) is organized based on these features. The main tool used for this is the self-organizing map (SOM) algorithm, which is a clustering method with intuitive visualization capabilities. In the process of developing this thesis the SOM has been applied several times to support the evaluation of the feature extraction process and the SOM is the basis of the final user interface to the music collection presented in this thesis.

In Section 4.1 the SOM is described briefly. Applying the SOM in Section 4.2 a brief evaluation of the features extracted in Chapter 3 is presented and discussed. Section 4.3 deals with different approaches to represent one piece of music based on the representation of its sequences. And finally in Section 4.4 the chapter is summarized.

4.1. Self-Organizing Maps

The goal of clustering data is to find groups (clusters) of data items that are similar to each other and different from the rest of the dataset. Clustering is a method to summarize the main characteristics of a dataset. For example, a music collection could be summarized by describing the groups (genres) it consists of. Each group could be described, for example, by the number and variation of its members. It might also be interesting to know the relationship between these groups. For example, a genre might have several sub-genres.

For the purpose of clustering data several algorithms have been developed. A recent review can be found in [JMF99]. One very frequently employed clustering algorithm is the SOM.

4.1.1. Background

The SOM was developed 1981 [Koh82] as an artificial neural network, which models biological brain functions. Since then it has undergone thorough analysis [Koh01]. The algorithm and its variations have been employed several times in domains such as machine vision, image analysis, optical character recognition, speech analysis, and engineering applications in general, see for example, [Oja92, IAD+92, KS97, KOS+96].

The SOM is a powerful tool that can be used in most data-mining processes [Ves00b] especially in data exploration [Kas97]. Moreover, the SOM is very efficient compared to other non-linear alternatives such as the Generative Topographic Mapping [BSW96], Sammon's Mapping [Sam69], or generally Multi Dimensional Scaling [KW78]. An ex-

ample for the efficiency of the SOM is the WebSOM[1] project [KKL+99].

4.1.2. The Batch SOM Algorithm

One of the variations of the original SOM is the batch SOM algorithm, which is significantly faster and has one parameter less to adjust [Koh92].

Although the algorithm is different the architecture of the map is the same. The map consists of map units, which are ordered on a grid. Usually this grid is rectangular and 2-dimensional and is used to visualize the data. An example can be seen in Figure 4.1(c). Each of the map units is assigned to a reference vector, also known as model or prototype vector. This vector lies in the data space and represents the data items that are closest to it. Units, which are close to each other on the grid, also have similar model vectors and thus represent similar data.

The batch SOM algorithm consists of two steps that are iteratively repeated until no more significant changes occur. First the distance between all data items $\mathbf{x}_i$ and the model vectors $\mathbf{m}_j$ is computed and each data item i is assigned to the unit that represents it best c_i. $\mathbf{V}_j$ denotes the Vernoi set of data items which are best represented by unit $\mathbf{m}_j$.

In the second step each model vector is adapted to better fit the data it represents. To ensure that each unit j represents similar data items as its neighbors, the model vector $\mathbf{m}_j$ is adapted not only according to $\mathbf{V}_j$ but also in regard to the Vernoi sets of the units in the neighborhood. Which units are considered to be neighbors and how much influence they have on the unit j is defined by a neighborhood function. A common choice for the neighborhood function $h_t(j,k)$ is a Gaussian function which is centered on $\mathbf{m}_j$ and has a standard deviation σ_t which decreases with each iteration t.

Assuming a Euclidean vector space, the two steps of the batch SOM algorithm can be formulated as

$$\begin{aligned} c_i &= \operatorname*{argmin}_j \|\mathbf{x}_i - \mathbf{m}_j\|^2, \quad \text{and} \\ \mathbf{m}_j^* &= \frac{\sum_i h_t(j,c_i)\,\mathbf{x}_i}{\sum_{i'} h_t(j,c_{i'})}, \end{aligned} \tag{4.1}$$

where $\mathbf{m}_j^*$ is the updated model vector. Although it is usually very unlikely it might occur that a data vector $\mathbf{x}_i$ is equally closest to two or more model vectors. In this case randomly one of these model vectors should be chosen to be c_i.

An efficient way to implement this is to first calculate the sum $\mathbf{s}_j$ of all data vectors in a Vernoi set $\mathbf{V}_j$ and then use them to calculate the weighted average,

$$\begin{aligned} \mathbf{V}_j &= \{\mathbf{x}_i |\, c_i = j\}, \\ \mathbf{s}_j &= \sum_{\mathbf{x}_i \in \mathbf{V}_k} \mathbf{x}_i, \quad \text{and} \\ \mathbf{m}_j^* &= \frac{\sum_k h_t(j,k)\,\mathbf{s}_j}{\sum_{k'} h_t(j,k')\,|\mathbf{V}_{k'}|}. \end{aligned} \tag{4.2}$$

[1] `http://websom.hut.fi/websom`

A Matlab® implementation which was used for this thesis can be found in Appendix A.1.1.

Parameters

Basically there are two parameters to adjust when using the batch SOM with a rectangular grid and a Gaussian neighborhood function, namely the map size and σ_t. Usually it is not difficult to find good parameter settings, and there are some basic rules of thumb that are helpful.

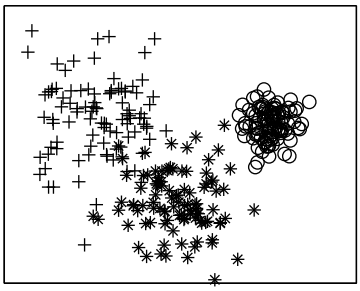
The *number of map units* should be somewhere in the range of $\sqrt{(}n)$, where n is the number of data items. The map size is usually rectangular. Depending on the display used (e.g. monitor or A4 paper), often a ratio between the width and the breadth of, for example, 4 to 3 is chosen. The number of units defines the resolution of the map. More units will lead to a higher resolution of the mapping. However, the SOM algorithm scales approximately $O(m^2)$, in regard to the number of units m.

The second parameter σ_t is also know as the *neighborhood radius*, since it defines in which radius of a unit other units are considered to be neighbors. In [KHKL96] well-established schedules are published which define at which iteration the radius should have which value. If the map is initialized well (e.g. using Principal Component Analysis) the initial radius can be set to about $c/8$, where c is either breadth or width of the map, choosing the bigger value. If a random initialization is chosen then a good initial radius is $c/4$. The radius then quickly decreases to $1/4$ of this initial value. This is known as the rough training phase. The actual number of iterations depends on the ratio between data items and units. The next and final phase is the fine tuning phase. Starting with the radius set to the value where the rough training ended the radius is slowly decreased to 0. When the neighborhood radius is 0 the SOM algorithm is the same as the k-means algorithm. With 0 neighborhood radius the units are only updated in regard to the data they best fit, ignoring surrounding units, and thus representing the data mapped to them as well as possible.

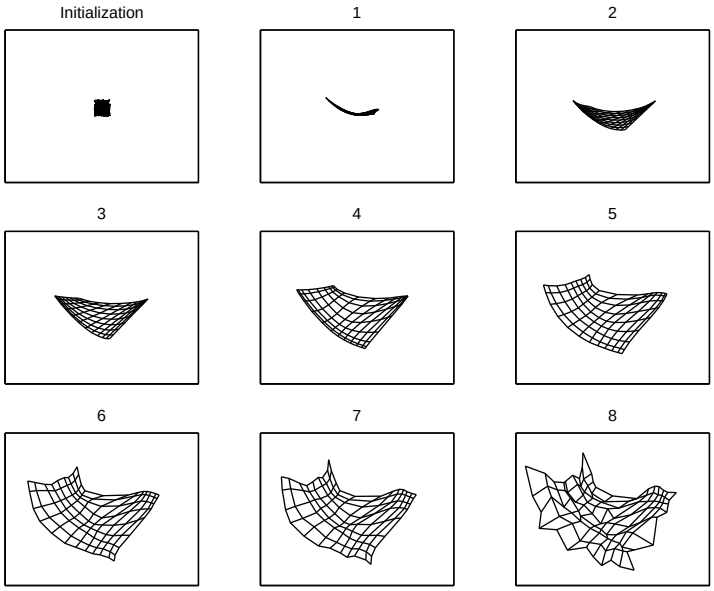
Simple Example

The algorithm is illustrated using Figure 4.1. The dataset (cf. Figure 4.1(a)) is 2-dimensional and contains three groups each represented by 100 members and normally distributed around their centers. Figure 4.1(b) depicts the form of the map in the data space. Starting with a random initialization, 8 subsequent training iterations are shown. After these 8 iterations the map fits the data rather well (cf. Figure 4.1(c)). There are several interesting aspects of the SOM that can be seen.

First of all the SOM does not map the data linearly. A linear mapping could be achieved using, for example, a Principle Component Analysis (PCA) [Hot33, Jol86]. The SOM uses its units efficiently, as few units as possible are wasted not representing any data. Furthermore, areas in the data space with a high density of data items



(a) Dataset



(b) Training

o	o	o	o	o	o	*		+	+	+	+
o	o	o	o	o	o		+	+		+	
o	o	o	o	o		*		+	+	+	+
o	o	o		*		*	+	+	+	+	+
o	o	*	*	*	*	*	+	+	+		+
*	*	*	*	*	*	*	+	+	+	+	+
*	*	*	*	*	*	+	+	+	+	+	+
*	*	*	*	*	*	*		+	+	+	+
*	*	*	*	*	*	+	+	+	+	+	+

(c) Map

Figure 4.1: The batch SOM algorithm illustrated with a simple 2-dimensional toy dataset. Starting with a random initialization 8 training iterations are calculated and the position of the model vectors in the data space visualized. Finally the units are labeled with the most frequent class they represent.

are represented by more units than areas with a lower density. This is also known as magnification [BSW97]. The model vectors are most dense in the area of the 'o' class (cf. Figure 4.1(b), 8th iteration).

Interesting is also the training itself. Notice how the map slowly unfolds to fit the

data. When the neighborhood radius is high initially, each map unit represents a large amount of data and thus the model vectors of the units tend to be around the mean of the whole dataset. As the neighborhood radius decreases the units become more flexible and are able to fit the data better. Notice also how the smoothness decreases drastically in the last two training iterations. These are the final steps of the fine tuning phase, regardless of the neighbors the units try to match their data as well as possible.

4.2. Music Sequences

The extracted features have been demonstrated in Chapter 3 using the first sequences of *Freak on a Leash* by *Korn*, *Für Elise* by *Beethoven*, *Rock DJ* by *Robbie Williams*, and *Yesterday* by the *Beatles*. In this section the clustering results of these 4 pieces of music and all their sequences are analyzed. Due to space limitations it is not possible to present a detailed evaluation of all 3940 sequences of the 359 songs, which are listed in Appendix B.

The sequences of the 4 songs can be seen in Figure 4.2. Generally, the sequences of one piece of music have similar characteristics. For example, one characteristic of the 9 sequences of *Für Elise* is that they have very low MFS values, while those of *Rock DJ* are about 10 times higher. Another rather typical characteristic is the beat pattern. For example, *Für Elise* hardly has any vertical lines, while *Rock DJ* has a few lines that are present in almost all of its 12 sequences. Another characteristic of *Rock DJ* is the strong but rather slow bass, which can be seen in most of its sequences in the lowest 2 frequency bands at a low modulation frequency (1.7Hz).

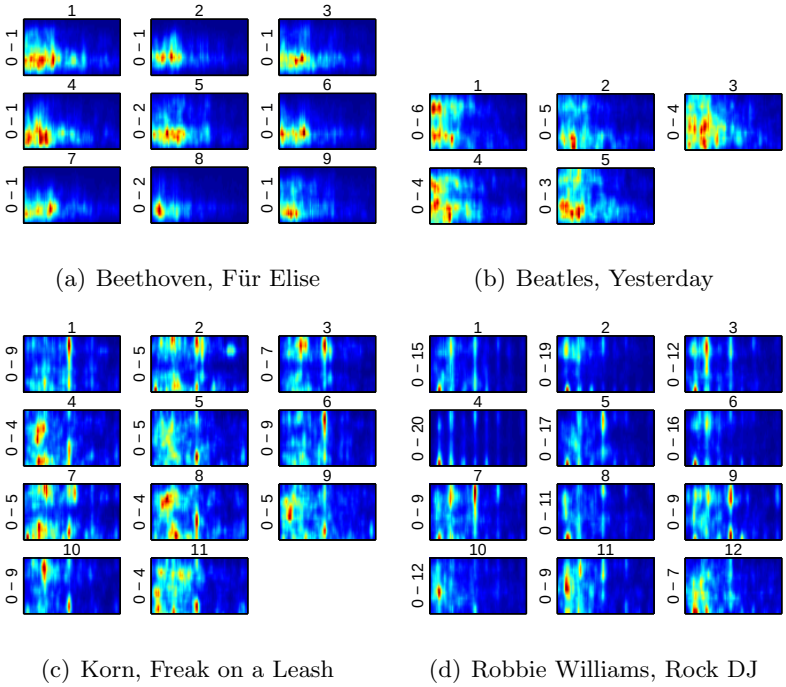
Figure 4.3 represents a 7x10 SOM trained with all 3940 sequences and labeled with the sequences of the 4 pieces of music. The clustering corresponds to the expected results. Almost all sequences of *Für Elise* are on one unit (upper right). The sequences of *Yesterday* are also close together (upper left). And there is a slight overlap (lower center) between *Freak on a Leash* and *Rock DJ*.

The same unit to which almost all sequences of *Für Elise* are mapped also maps almost all sequences of *Moonlight Sonata* by *Beethoven* and almost all sequences of *Eine kleine Nachtmusik* by *Mozart*. There are a total of 145 sequences mapped to this unit and all are rather peaceful classical music.

The unit that represents the sequences 1 and 4 of *Yesterday* represents a total of 83 sequences. For example, 11 sequences of the songs, *Adia*, *Angel*, and *I will remember you* by *Sarah McLachlan*, 5 sequences of *American Pie* by *Don McLean*, and 4 sequences of *I believe I can Fly* by *R Kelly*, are mapped to this unit.

The unit that represents the sequences 1, 3, 4, 5, 6 of *Rock DJ* represent a total of 26 sequences. For example, it represents almost all sequences of *Coco Jambo* by *Mr President* and 5 sequences of *N2gether now* by *Limp Bizkit*, which is one of the less aggressive songs of *Limp Bizkit*.

The unit, which represents most of the *Freak on a Leash* sequences, namely the



(a) Beethoven, Für Elise (b) Beatles, Yesterday

(c) Korn, Freak on a Leash (d) Robbie Williams, Rock DJ

Figure 4.2: The 6-second music sequences from 4 pieces of music. The images represent the MFS diagrams presented previously.

sequences 4, 5, 7, 8, 9, and 11, represents a total of 45 sequences. For example, 12 sequences of 9 different songs by *Limp Bizkit*, and all 9 sequences of *Love is in the air* by *John Paul Young*. This unit appears to be very inhomogeneous in respect to the mapped sequences. It is not clear why *Love is in the air* is mapped together with *Freak on a Leash*, their MFS patterns are rather different, however, they have a similar range of values.

More insight into the map can be obtained by analyzing the units and the prototype sequences they represent. Figure 4.4 illustrates the relative, and Figure 4.5 the absolute model vectors. The former figure reveals details that are not visible if all model vectors are represented mapping the absolute values to the same color scale. However, it is important to keep the absolute values in mind. For example, the units (8,7), (9,7) and (10,7) look very similar in Figure 4.4 apparently having the same patterns, while Figure 4.5 reveals that there is a significant difference. The MFS values of (10,7) are higher then those of (9,7), which are higher than those of (8,7). On the other hand the units (1,6) and (1,7) in Figure 4.5 seem to be identical while Figure 4.4 reveals that the difference is that (1,6) has activities in the lower and upper critical-bands while (1,7) only has activities in the lower bands.

Notice the similarities between neighboring units, which is the result of the neighborhood function of the SOM. It is interesting that between map areas with rather specific patterns of vertical lines there are units with hardly any clear lines. For example, in Figure 4.4 between the area around unit (9,7) and the area around unit (5,4) there is a set of units (8,5), (7,5), (7,6), and (6,6) which mark the boarder. This is caused

						elise.1 elise.2 elise.3 elise.4 elise.6 elise.7 elise.8
	yesterday-b.2	yesterday-b.5				
yesterday-b.3	yesterday-b.1 yesterday-b.4					elise.9 elise.5
				korn-freak.2		
rockdj.10 rockdj.11					korn-freak.4 korn-freak.5 korn-freak.7 korn-freak.8 korn-freak.9 korn-freak.11	
				rockdj.12		
			korn-freak.3 korn-freak.10			
	rockdj.2		korn-freak.1 korn-freak.6 rockdj.7 rockdj.8 rockdj.9			
		rockdj.1 rockdj.3 rockdj.4 rockdj.5 rockdj.6				

Figure 4.3: A SOM trained with all 3940 sequences where the units are labeled with the sequences of the 4 songs.

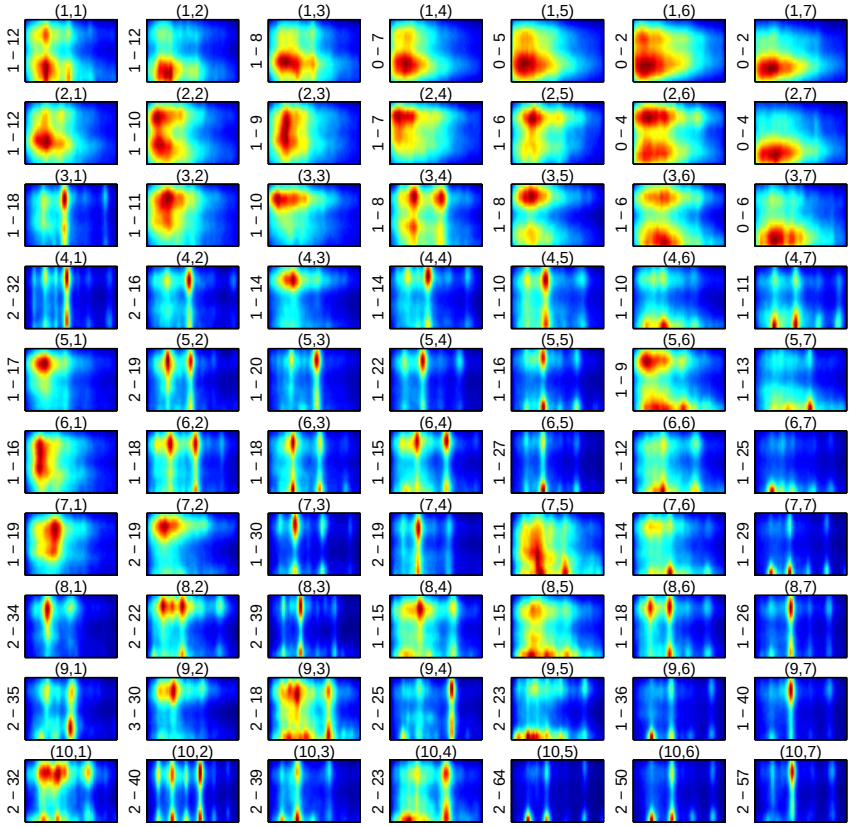


Figure 4.4: The model vectors of the SOM in Figure 4.3. The model vectors are represented as MFS diagrams in the dimensions of critical-band rate (vertical) vs. modulation frequency (horizontal) and modified fluctuation strength (color coding). The color coding of each model vector is scaled to the minimum and maximum value of the specific model vector. The values to the left of the each subplot indicate the range of the values scaled to the range of 0 to 64 in respect to the maximum and minimum of the all model vectors.

by the point-wise comparison. The average of two significantly different patterns of vertical lines might result in the loss of these lines. Notice that most of the sequences of *Freak on a Leash* are mapped to one of these intermediate units (6,6). Comparing the sequences (cf. Figure 4.2(c)) and the model vector of the unit (6,6) yields that the main similarity is the range of the values. The patterns of the sequences of *Freak on a Leash* are not as smooth as their unit would suggest.

The main reason for this is that there are not enough units available to represent all the sequences with their details. Frühwirth [Frü01] has used a SOM with 22 by 22 map units to cluster the segments. Using similar map sizes leads to model vectors with more details as units represent fewer sequences and thus can adapt better to them. However, the sequences of one piece of music are then no longer represented by so few units and tend to spread out more.

All in all the extracted features enable clustering algorithms to organize music according to a criteria that correspond to some degree with the concept of music genres.

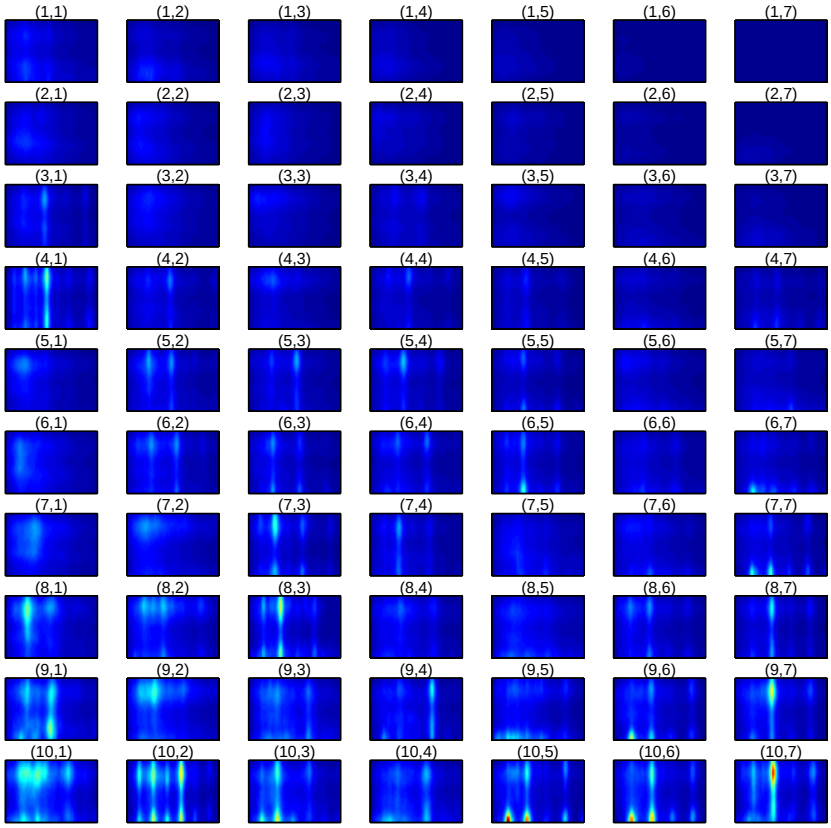


Figure 4.5: The model vectors of the SOM in Figure 4.3. Unlike Figure 4.4 the color coding is not scaled to each model vector instead it is scaled to the all model vectors. Thus the color coding corresponds to the way the SOM algorithm sees the model vectors.

4.3. Pieces of Music

This section deals with the question of how to represent a piece of music based on its 6-second sequences. This is necessary to obtain the final map of the whole pieces of music and not their sequences as in the previous section. The representations should be in a vector space and the metric, which defines the distance between the vectors of the pieces of music, should correspond to their perceived similarity.

One straightforward approach would be to use the average of the sequences to represent to corresponding piece of music. On the other hand it seems to be very intuitive that most pieces of music will contain different types of sequences. For example, a piece of music might include some slower sequences and some faster sequences. Also the instruments used and the existence of vocals may alternate between the different sequences of one piece of music. It is questionable if the average of a slow and a fast sequence would be a good representation for such a piece of music.

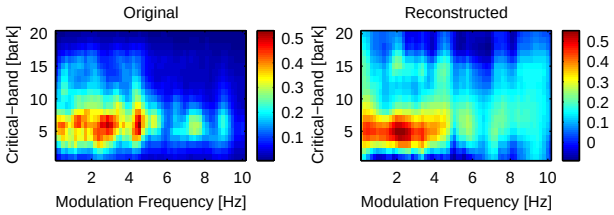
To decide how to combine the sequences four methods are evaluated using the SOM. The following sub-sections start with a description of Principal Component Analysis, which is used throughout this section. Then the different methods are described and evaluated. The methods are Gaussian Mixture Models, Fuzzy C-Means, k-means with a simple response measure, and finally the simplest approach, the median of the sequences.

4.3.1. Principal Component Analysis

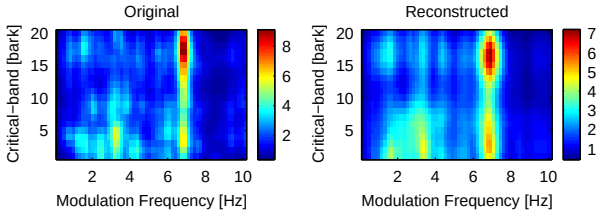
In many data analysis applications such as image recognition, e.g. [LB96], it is common to use Principal Components Analysis (PCA) [Hot33, Jol86] to reduce the dimensionality of the data. Especially, when the data is highly correlated, such as the MFS representations of the music sequences, the d-dimensional data can be compressed very efficiently using PCA and resulting in a much lower m-dimensional *eigenspace*.

The PCA is a linear projection, and it guarantees that the m-dimensional eigenspace best preserves the variance of the data in the d-dimensional space. This is equivalent to preserving the sum of the squared distances between the data points in the d-dimensional space in the m-dimensional space as well as possible. Often this is referred to as minimizing the *reconstruction error* or minimizing the empirical risk.

The MFS data is represented in 1200 dimensions, corresponding to first 20 critical-bands and 60 levels of the modulation frequency in the range from 0-10Hz. These 1200 dimension can be reduced significantly using PCA while preserving most of the information. Figure 4.6 depicts how much of the information is preserved using 20 dimensions, Figure 4.7 depicts the effects of using 80 dimensions. As can be seen using only 20 dimensions, which is a reduction by a factor of 60, yields reasonable good representations. The main characteristics are preserved, for example, it is clear that the values are very low and there are no vertical lines in *Für Elise*, while the values of *Freak on a Leash* are much higher and there is a significant vertical line below 7Hz. Using 80 dimensions sharpens the images so that most details become apparent, for



(a) Beethoven, Für Elise



(b) Korn, Freak on a Leash

Figure 4.6: Comparison between the 1200-dimensional data space and the 20-dimensional eigenspace. On the left side are the original images, on the right side are the compressed and decompressed images, which are first projected into the 20-dimensional eigenspace and then reconstructed (decompressed) using the eigenvectors in the 1200-dimensional data space.

example the low valued, vertical lines of *Freak on a Leash* around 2Hz to 4Hz.

There are several advantages when using low-dimensional representations. Obviously, less storage is required and the clustering algorithms need fewer computations to find solutions. Furthermore, clustering algorithms which use random elements, for example for initialization, will more likely produce similar results in different runs, since the number of possible solutions and the number of variables which need to be determined are decreased significantly. All in all the benefits justify the loss of some details.

The linear projection $\mathbf{x}^*$ into the low-dimensional eigenspace of a vector $\mathbf{x}$ is calculated as

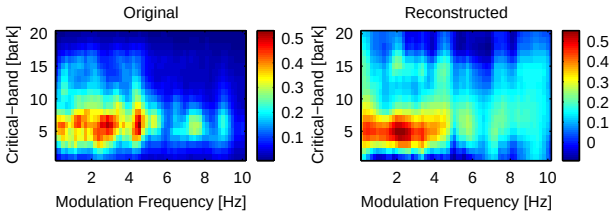
$$\mathbf{x}^* = (\mathbf{x} - \mathbf{c})\mathbf{V}, \tag{4.3}$$

where $\mathbf{V}$ is a d by m matrix with orthonormal columns and $\mathbf{c}$ is the mean of all vectors in the dataset. The vector $\mathbf{x}$ is reconstructed from $\mathbf{x}^*$ with

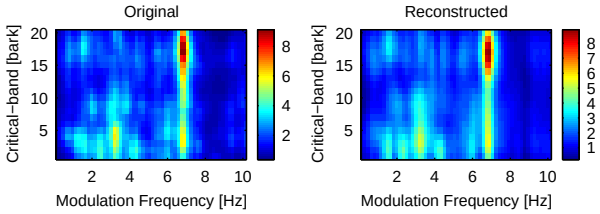
$$\mathbf{x} = \mathbf{x}^*\mathbf{V}^T + \mathbf{c}. \tag{4.4}$$

The m columns of the matrix $\mathbf{V}$ are the first m *eigenvectors* of the covariance matrix $\mathbf{Q}$ sorted descending according to their *eigenvalues*. The d by d matrix $\mathbf{Q}$ is calculated from the dataset after $\mathbf{c}$ has been deducted, so that the data has zero mean. The space spanned by the m orthonormal eigenvectors is referred to as the eigenspace. The matrix $\mathbf{V}$ can be computed efficiently based on *Singular Value Decomposition* (SVD) [MN95].

The first 20 eigenvectors of the music dataset can be seen in Figure 4.8. The first eigenvector represents music without a strong beat reoccurring in exactly the same



(a) Beethoven, Für Elise



(b) Korn, Freak on a Leash

Figure 4.7: Comparison between the 1200-dimensional data space and the 80-dimensional eigenspace. The same as in Figure 4.6, except that the images are reconstructed from the 80-dimensional eigenspace instead of the 20-dimensional eigenspace.

intervals, such as classical music. The second eigenvector represents music with a strong bass with about 120bpm and a double as fast beat across all frequencies. The third eigenvector represents similar music without a bass. Notice that the green colors are 0 values and correspond to default values. If an eigenvector represents a feature which is beyond the average then this corresponds to red colors and if the eigenvector lacks an average feature this is represented with blue colors. For example, the third eigenvector has blue colors around the 120bpm (2Hz) bass. Note that the coefficients, which define by which eigenvectors a data vector is represented, can be negative. This applies for example to classical music, which is mainly represented by the first eigenvector. The lengths of the eigenvectors are normalized to one.

4.3.2. Gaussian Mixture Models

As stated previously, the goal is to represent the pieces of music based on their sequences. One possibility is to assume that the music sequences represent only a fraction of all available music sequences. Furthermore, it is straightforward to assume that there are different clusters of sequences and that most sequences cannot clearly be assigned to one specific cluster but rather have a certain probability of belonging to each cluster. A piece of music can then be defined by the average probability by which its sequences belong to the clusters. Finally it is necessary to make some assumptions regarding the shape of the clusters. It is very common to assume Gaussian clusters so that each cluster is defined through a center and a covariance matrix.

These assumptions lead to Gaussian mixture models (GMM) [DH73, Bis95]. The GMM models the density function that generates the dataset from a mixture of Gaussian

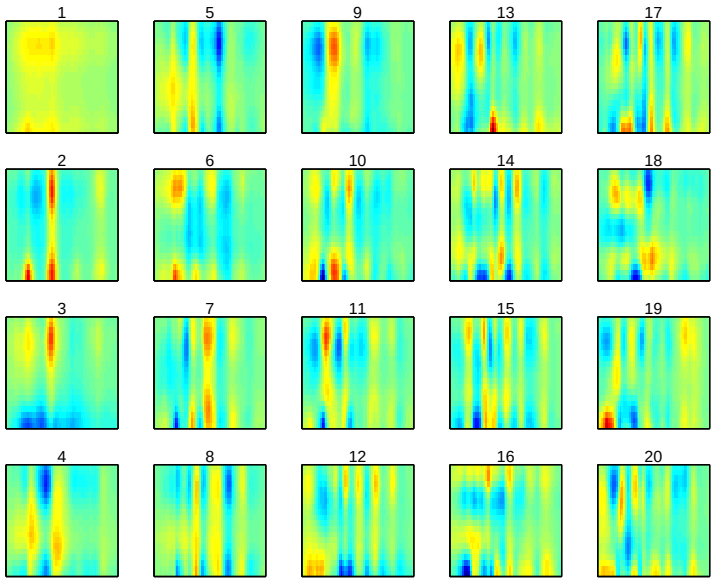


Figure 4.8: The first 20 eigenvectors with the highest eigenvalues.

basis functions (clusters). The (posterior) probability $P(i|\mathbf{x})$ that a data item $\mathbf{x}$ is generated by a cluster i is

$$P(i|\mathbf{x}) = \frac{P(\mathbf{x}|i)P(i)}{P(\mathbf{x})}, \tag{4.5}$$

where $P(i)$ is the prior probability of the cluster i and $\sum_i P(i) = 1$. The normalization factor $P(\mathbf{x}) = \sum_i P(\mathbf{x}|i)$ ensures that $\sum_i P(i|\mathbf{x}) = 1$. The likelihood $P(\mathbf{x}|i)$ to observe a data item $\mathbf{x}$ in a cluster i is given by the multi-dimensional normal distribution

$$P(\mathbf{x}|i) = \frac{1}{(2\pi)^{d/2}\det(\boldsymbol{\Sigma}_i)^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_i)^T\boldsymbol{\Sigma}_i^{-1}(\mathbf{x}-\boldsymbol{\mu}_i)\right), \tag{4.6}$$

where d is the dimensionality of the data space. For a cluster i the center is denoted by $\boldsymbol{\mu}_i$, $\boldsymbol{\Sigma}_i$ denotes the covariance matrix and $\det(\boldsymbol{\Sigma}_i)$ its determinant.

The likelihood to observe the dataset $\mathbf{X}$ given the GMM defined by the set of parameters $\boldsymbol{\theta}$ is

$$\ell(\boldsymbol{\theta}) \equiv P(\mathbf{X}|\boldsymbol{\theta}) = \prod_{n=1}^{N} P(\mathbf{x}_n|\boldsymbol{\theta}). \tag{4.7}$$

The parameters of a GMM can efficiently be estimated iteratively using the EM algorithm [DLR77]. The EM algorithm consists of two steps, namely the *expectation* and the *maximization* step which are repeated iteratively until the likelihood function $\ell(\boldsymbol{\theta})$ converges to a local maximum. For details on the EM algorithm see, for example [Bis95, Rip96].

For the experiments with the music collection the GMM-EM functions of the Matlab® Netlab[2] toolbox [Nab01] were used. Due to numerical problems the eigenspace with only 20 eigenvectors was used and the number of clusters was limited to 30. To allow

[2]http://www.ncrg.aston.ac.uk/netlab/

the clusters to fit the data better the full covariance matrix was estimated, which means that the clusters could be any form of ellipsoid rotated in any direction.

To initialize the covariance matrices and the centers of the basis functions 5 iterations of k-means were used. And a few GMM-EM runs were made from which the model with the best likelihood was selected.

The 30-dimensional vector $\mathbf{u}$ representing a piece of music with the indices of its sequences given by M is calculated as

$$\mathbf{u}(i) = \frac{1}{|M|} \sum_{m \in M} P(i|x_m), \quad \text{for } i = 1 \dots 30. \tag{4.8}$$

Note that $\sum_i \mathbf{u}(i) = 1$.

Typical results of training a SOM based on vectors of pieces of music combined from their sequences by GMM-EM can be seen in Figure 4.9. Because of the low-dimensionality of the sequence eigenspace and the low number of clusters for the GMM, the results are very stable and runs with similar parameters but (randomly chosen) different initializations will yield very similar results.

The map unit, which represents *elise* (*Für Elise* by *Beethoven*), usually always represents the same pieces of music which are *adagio* (*Adagio from a clarinet concert* by *Mozart*), *air* (*Air from the orchestrasuite #3* by *Bach*), *kidscene* (*Fremde Länder und Menschen* by *Schuhman*), and *mond* (*Mondscheinsonate* by *Beethoven*). All of these are slower, playful, peaceful, classical pieces of music with few instruments, for example, *kidscene* is a piano solo, and air includes only a few string instruments.

Usually right next to this unit will be units representing *fortuna* (*O Fortuna Imparix Mundi by Carl Orff*), *funeral* (*Funeral march* by *Chopin*), *avemaria* (*Ave Maria* by *Schubert*), *nachtmusik* (*Eine kleine Nachtmusik* by *Mozart*), *schindler* (*Schindlers List* by *John Williams*), and *jurasicpark* (*Jurassic Park* by *John Williams*). These are also classical, however, they appear to be more dynamic and more powerful. Generally, all classic music in the collection on the map is located around the lower left to the center left.

Not as clear as the mapping of *elise* is the mapping of *korn-freak* (*Freak on a Leash* by *Korn*). The song is usually not located together with exactly the same but always around *limp-pollution* (*Pollution* by *Limp Bizkit*), *limp-nobodyloves* (*Nobody loves me* by *Limp Bizkit*) and other songs from *Limp Bizkit*. Sometimes the perceptually very different songs *deepisyourlove* (*How deep is your love* by *Take That*), and *wildwildwest* (*Wild Wild West* by *Will Smith*) are mapped near by too. The map does not seem to distinguish very well between songs like *korn-freak*, and *Limp Bizkik*, *Papa Roaches*, *New Model Army* songs in general on one side, and songs like *lastchristmas* (Last Christmas by *Wham*) and *deepisyourlove* on the other side. This might be caused by the limitations regarding the dimensionality of the eigenspace and the limited number of clusters used to represent the sequences.

The song *rockdj* (*Rock DJ* by *Robbie Williams*) is always around *limp-counterfeit*, *limp-nookie* and other *Limp Bizkit* songs and usually also next to *cocojambo* (*Coco*

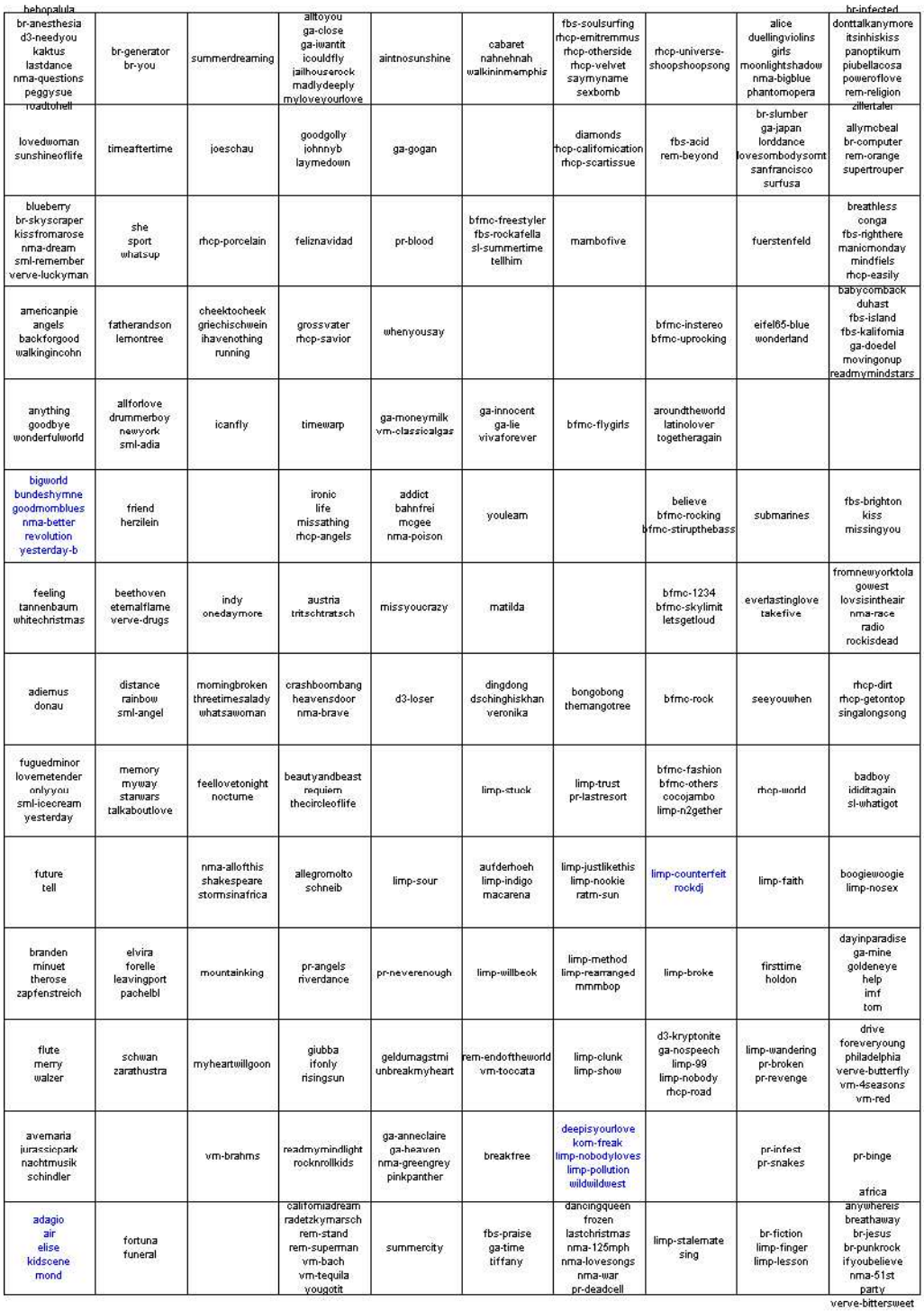


Figure 4.9: The SOM trained with the pieces of music represented based on the GMM-EM approach. The blue highlighted map units are the units which represent the 4 pieces of music that are analyzed in detail. The labels are identifiers to which the full title and interpret or author can be found in Appendix B.

Jambo by *Mr. President*). All these songs have stronger beats at about the same frequency as Rock DJ and vocals. However, there seems to be a significant perceptual difference between the rather aggressive *Limp Bizkit* songs and *Rock DJ*.

The song *yesterday-b* (*Yesterday* by the *Beatles*) is always co-located with *bundeshymne* (the *national anthem* of Austria by *Mozart*), *goodmorningblues* (*Good Morning Blues* by *Frank Muschalle*), *nma-better* (*Better than them* by *New Model Army*), *revolution* (*Do you hear the people sing?* from *Les Mirables*), sometimes together with *whitechristmas* (*White Christmas* by *Bing Crosby*).

All in all the results are not completely satisfactory, which might be related to the limitations caused by the numerical problems of the statistically sound GMM-EM.

4.3.3. Fuzzy C-Means

An alternative to the GMM is Fuzzy C-Means (FCM) [Bez81, CM98], which basically offers the same solution in regard to combining sequences of a piece of music to one vector.

The goal of FCM is to find the fuzzy cluster centers and the values of fuzzy membership of the data items to the clusters, minimizing the following error function,

$$E = \sum_{i=1}^{C} \sum_{n=1}^{N} \rho(i, \mathbf{x}_n)^b \left\| \mathbf{x}_n - \boldsymbol{\mu}_i \right\|^2, \tag{4.9}$$

where the C is the number of clusters, N is the number of data items (music sequences), $\boldsymbol{\mu}_i$ is the center of the i-th cluster, $\mathbf{x}_n$ is the n-th data vector and the paramter $b > 1$ is a fixed value, which needs to be specified. When $b = 1$ the error function is equivilant to the error function of k-means. For $b \to \infty$ all cluster centers converge to the centroid of the training dataset. In other words, the clusters become completely fuzzy so that each data point belongs to every cluster to the same degree. Typically the value b is chosen around 2 [CM98].

Additionally to the error function FCM uses the following constraints,

$$\sum_{i=1}^{C} \rho(i, \mathbf{x}_n) = 1, \quad \text{for } i = 1, \ldots, N, \tag{4.10}$$

which ensures that the membership of any data item to all clusters adds up to 1.

The FCM algorithm minimizes the error function (Equation 4.9) subject to the constraints (Equation 4.10). A Matlab® implementation can be found in Appendix A.1.2..

The iterative algorithm consists of two steps which are repeated until the error function converges. First the memberships are (re-)estimated based on some estimation for the cluster centers, and then the cluster centers are re-estimated based on the new

values for the membership. The respective calculations are,

$$\boldsymbol{\mu}_i = \frac{\sum_i \rho(i,\mathbf{x}_n)^b \mathbf{x}_n}{\sum_i \rho(i,\mathbf{x}_n)^b}, \quad \text{and} \tag{4.11}$$
$$\rho(i,\mathbf{x}_n) = \frac{z_{in}}{\sum_{k=1}^{C} z_{kn}}, \quad \text{where } z_{in} = \|\mathbf{x}_n - \boldsymbol{\mu}_i\|^{2/(1-b)}.$$

Typical results when using FCM can be seen in Figure 4.10. For the experiments with the music collection the 80 dimensional eigenspace of the music sequences was used, the centers of the FCM were initialized using a few iterations of the faster k-means algorithm, the number of clusters was set to 120 and the parameter b was set to 1.1 which is rather low. The reason for the low value of b is that higher values yielded very similar cluster centers which did not represent the variations in the data appropriately.

The higher dimensionality of the eigenspace and the higher number of clusters dramatically increases the number of possible solutions to the clustering problem. Due to the random initialization of the FCM cluster centers (more accurately, the random initialization of the k-means centers, which are used to initialize the FCM centers) there is a greater variation between different FCM results then between the GMM-EM results.

The unit, which represents *elise* (*Für Elise* by *Beethoven*), usually also represents *kidscene* (*Fremde Länder und Menschen* by *Schuhmann*) and *mond* (*Mondscheinsonate* by *Beethoven*) and is always near *merry* (*The Merry Peasant* by *Schuhmann*), *adadio* (*Adagio from a clarinet concert* by *Mozart*), *air* (*Air from the orchestrasuite #3* by *Bach*), *fortuna* (*O Fortuna Imparix Mundi* by *Carl Orff*), *funeral* (*Funeral march* by *Chopin*), and other similar classical pieces of music. Generally, all classic music in the collection on the map is located around the lower right to the lower center.

For *korn-freak* (*Freak on a Leash* by *Korn*) the FCM results are slightly better compared to those using GMM-EM, because the 4 times higher number of eigenvectors and clusters allows a higher resolution. Frequently similar songs by *Papa Roaches* (such as *pr-revenge* and *pr-deathcell*) and several songs by *Limp Bizkit* are located around *korn-freak*, however, the song *wildwildwest* (*Wild Wild West* by *Will Smith*) is still in the immediate vicinity.

The unit representing *rockdj* (*Rock DJ* by *Robbie Williams*) is usually always located together with different pieces of music when comparing two FCM runs with the same parameters but different random initialization. In Figure 4.10 *rockdj* is located together with *breakfree* (*I want to break free* by *Queen*), *nahnehnah* (*Nah Neh Nah* by *Vaya Con Dios*), and *shoopshoopsong* (*Shoop Shoop Song* by *Cher*). Sometimes *rockdj* might be mapped as the only piece of music to a specific unit and sometimes *rockdj* is mapped together with songs like *cocojambo* (*Coco Jambo* by *Mr. President*), *submarine* (*Yellow Submarine* by the *Beatles*), *sexbomb* (*Sexbomb* by *Tom Jones*), or *bongobong* (*Bongo Bong* by *Manu Chao*).

The unit representing *yesterday-b* (*Yesterday* by the *Beatles*) usually also represents *goodbye*, *philadelphia*, *she*, *youlearn* and sometimes *sunshineoflife* (*You are the sunshine*

Figure 4.10: The SOM trained with the pieces of music represented based on the FCM approach. The blue highlighted map units are the units which represent the 4 pieces of music that are analyzed in detail. The labels are identifiers to which the full title and interpret or author can be found in Appendix B.

of my life by *Steve Wonder*), *kissfromarose* (*Kiss from a rose* by *Seal*), *wonderfulworld* (*What a wonderful world* by *Louis Armstrong*), and *sml-remember* (*I will remember you* by *Sarah McLachlan*). All of these are somehow similar.

4.3.4. K-Means

As mentioned previously the k-means algorithm [Mac67] has been used to initialize the FCM and GMM-EM algorithms. The k-means algorithm basically does the same (crisp clustering) as the SOM, with the significant difference that there is no neighborhood defined between two clusters (map units). Practically, k-means is equivalent to the SOM when the neighborhood radius of the SOM is set to zero.

The k-means algorithm can be used in the same way as the FCM and GMM-EM algorithms, however, each music sequence will not have a fuzzy membership or probability of belonging to several cluster but instead will be assigned to one cluster only.

The error function the k-means algorithm minimizes is

$$E = \sum_{n=1}^{N} \|\mathbf{x}_n - \boldsymbol{\mu}_{c(n)}\|^2, \tag{4.12}$$

where $\boldsymbol{\mu}_{c(n)}$ is the centroid of the cluster closest to the data vector x_n. A Matlab® implementation can be found in Appendix A.1.3.

The centers of the clusters are initialized randomly, for example, by selecting C random data items. Then two steps are repeated iteratively until the error function (Equation 4.12) converges. The first step is to find $\boldsymbol{\mu}_{c(n)}$ for all $n = 1, \ldots, N$ and in the second step for each cluster the new center is calculated as the mean of all data items belonging to it.

The experiments with the music collection were conducted setting the number of clusters to 120. Several runs were made and based on the error function the best clustering was chosen. Since each piece of music is usually represented by more than one sequence counting the average number of sequences per cluster would yield fuzzy results. The degree of fuzziness can be increased, for example, using a very simple method such as giving the best matching cluster for a sequence 3 points, the second best 2 points and the third best 1 point. Then the average points of all clusters can be calculated for each piece of music.

A typical result of this approach can be seen in Figure 4.11. Similar to the FCM results also here the results depend on the random initialization and thus two runs might produce different results.

However, the classical music cluster is usually always the same. Especially, *elise* (*Für Elise* by *Beethoven*) is always around the same group, which is basically the same as the group which can be seen in the GMM-EM or FCM results. Interesting though is that the *mountainking* (*In the hall of the mountainking* by *Grieg*) is mapped to the same unit as *elise*. The two and a half minutes of *mountainking* are represented by 6 6-second sequences. The first 3 are very quiet, then gradually the loudness rises and

aroundtheworld believe eifel65-blue latinolover togetheragain wonderland	bfmc-instereo bfmc-rocking bfmc-stiruptheba bfmc-uprocking letsgetloud	bfmc-1234 bfmc-freestyler kiss seeyouwhen sl-summertime		aintnosunshine alltoyou goodgolly jailhouserock johnnyb madlydeeply myloveyourlove tellhim	cocojambo limp-n2gether rockdj singalongsong	bfmc-others bfmc-rock limp-indigo limp-justlikethis macarena	bfmc-fashion limp-broke limp-counterfeit limp-faith limp-method limp-nookie limp-stuck	limp-finger limp-show limp-sour	breathaway br-punkrock d3-kryptonite goldeneye help pr-lastresort pr-snakes wildwildwest
bfmc-skylimit fbs-kalifornia	bfmc-flygirls conga	rhcp-dirt rhcp-getontop		pr-blood	mambofive	limp-willbeok	limp-clunk limp-pollution limp-trust pr-infest rockisdead	limp-99 limp-nobody limp-rearranged	aufderhoeh imf tom
dschinghiskhan themangotree	fbs-soulsurfing	rhcp-californication rhcp-world sl-whatigot	ga-gogan ratm-sun saymyname	badboy rhcp-savior summerdreaming	party verve-bittersweet	limp-lesson	limp-stalemate	korn-freak limp-nobodyloves pr-deadcell pr-neverenough pr-revenge rem-endoftheworld	africa limp-wandering
rhcp-easily rhcp-emitremmus rhcp-scartissue rhcp-universe- rhcp-velvet	mindfiels	bongobong mmmbop	walkininmemphis	fbs-rockafella feliznavidad ga-close	ifyoubelieve	anywhereis ididitagain	ga-time holdon limp-nosex pr-broken	pr-binge	br-generator br-you verve-butterfly vivaforever vm-4seasons vm-red
	itsinhiskiss rhcp-otherside sexbomb	shoopshoopsong	cabaret nahnehnah	boogiewoogie diamonds running submarines	rhcp-road	deepisyourlove nma-125mph		br-jesus ironic nma-51st	drive
fromnewyorktola gowest lovsisintheair radio	everlastinglove	ga-nospeech tiffany	fbs-brighton missingyou	lastchristmas	dayinparadise ga-mine	vm-toccata	addict rhcp-angels timewarp	br-anesthesia nma-poison vm-classicalgas	br-fiction ga-iwantit icouldfly laymedown
fuerstenfeld girls	panoptikum rem-orange	br-computer ga-japan sanfrancisco	piubellacosa	breakfree rem-beyond sing	unbreakmyheart	cheektocheek d3-loser	whenyousay	ga-innocent	ga-moneymilk
duellingviolins manicmonday nma-bigblue	br-infected	poweroflove zillertaler	donttalkanymore	fbs-praise nma-war	firsttime ga-anneclaire ga-heaven	ga-lie rhcp-porcelain	backforgood youleam	allforlove grossvater	d3-needyou griechischwein joeschau lastdance lemontree lovedwoman
allymcbeal breathless moonlightshadow rem-religion	alice	matilda	dingdong	veronika	philadelphia	sml-remember yesterday-b	newyork sml-adia wonderfulworld	goodbye she	kaktus nma-lovesongs takefive
babycomback supertrouper surfusa	duhast ga-doedel movingonup	fbs-island readmymindstars	frozen		drummerboy starwars verve-drugs	austria readmymindlight	herzilein icanfly lovesombodysomt	americanpie kissfromarose	blueberry br-skyscraper nma-dream peggysue sunshineoflife verve-luckyman
	fbs-righthere	lorddance riverdance		goodmornblues	feeling myheartwillgoon talkaboutlove	friend	anything ifonly missathing	walkingincohn	bebopalula br-slumber nma-race sport timeaftertime whatsup
branden zapfenstreich	zarathustra	beautyandbeast future	nocturne shakespeare threetimesalady	bundeshymne morningbroken sml-angel	distance whitechristmas	nma-better	angels bigworld heavensdoor ihavenothing whatsawoman	crashboombang missyoucrazy nma-questions	roadtohell
adagio air avemaria flute merry nachtmusik	forelle minuet schindler tell walzer	schwan therose		onlyyou schneib yesterday	donau myway rainbow	memory nma-brave	eternalflame fbs-acid	fatherandson mcgee	dancingqueen foreveryoung geldumagstmi rocknrollkids vm-tequila yougotit californiadream
elise fortuna funeral kidscene mond mountainking	jurassicpark	elvira leavingport pachelbl stormsinafrica	beethoven indy	adiemus allegromolto feellovetonight lovemetender requiem	fuguedminor sml-icecream vm-bach	bahnfrei nma-allofthis pr-angels thecircleoflife tritschtratsch	giubba life onedaymore revolution tannenbaum vm-brahms	nma-greengrey	phantomopera pinkpanther radetzkymarsch rem-stand rem-superman risingsun summercity

Figure 4.11: The SOM trained with the pieces of music represented based on the k-means approach. The blue highlighted map units are the units which represent the 4 pieces of music that are analyzed in detail. The labels are identifiers to which the full title and interpret or author can be found in Appendix B.

reaches its maximum in the 6th sequence, which is significantly louder than all others. While the first 5 sequences are well comparable to *elise*, the 6th sequence is perceptually very different. The reason why *mountainking* and *elise* are mapped together lies in the ranking method used, which is very robust in respect to outliers.

The unit representing korn usually also represents *pr-revenge*, *pr-deathcell*, *limp-nobodyloves*, *pr-binge*, and other songs by *Papa Roaches* and *Limp Bizkit*, which match rather well. A song which does not seem to match too well is *rem-endoftheworld* (*Its the end of the World* by *REM*). The songs of *REM* in general seem to be much less aggressive then those of *Papa Roaches* or *Limp Bizkit*, however, the specific song *Its the end of the world*, is more aggressive then the usual *REM* songs.

The unit representing *rockdj* (*Rock DJ* by *Robbie Williams*) usually also represents *macarena* (*Macarena* by *Los Del Rio*), *cocojambo* (*Coco Jambo* by *Mr. President*), *bongobong* (*Bongo Bong* by *Mau Chao*), *singalongsong* (*Sing along song* by *Tim Tim*) and *limp-n2gether* (*N2gether now* by *Limp Bizkit*), which match well.

The unit representing *yesterday-b* (*Yesterday* by the *Beatles*) usually also represents *wonderfulworld* (*What a wonderful world* by *Louis Armstrong*), *sml-adia* (*Adia* by *Sarah McLachlan*), and *newyork* (*New York, New York* by *Frank Sinatra*), which match well too.

4.3.5. Median

So far the approaches assumed that the sequences of a piece of music might have perceptually significant differences. Alternatively, it is possible to assume that the differences between the sequences of a song are perceptually insignificant and that only the parts they have in common are significant. To obtain a common representation of several sequences the mean of all characteristics can be calculated. However, the mean is sensitive to outliers and thus the median seems to be more appropriate, especially considering that less then 1/3 of all available sequences are used to represent a piece of music. For example, *In the hall of the mountainking* by *Grieg* is represented by 5 rather quite and one very powerful sequence. The one powerful sequence would dominate the mean, while it has practically no influence on the median.

For the experiments with the music collection the median is calculated from the 1200-dimensional MFS values, and then represented in the 80-dimensional eigenspace. The median of *Für Elise*, *Freak on a Leash*, *Yesterday*, and *Rock DJ* can be seen in Figure 4.12. Comparing it to Figure 4.2 reveals the effects of calculating the median.

The median of *Für Elise* summarizes the sequences very well. The median indicates that there are activities in the range of 3-10 bark and with a modulation frequency of up to 5Hz. However, the values of these activities are very low. The single sequences of *Für Elise* have many more details, for example, the first sequence has a minor peek around bark 5 and 5Hz modulation frequency.

The median of *Yesterday* is a good summary as well. There are several activities in lower and some in the upper critical-bands and there is no significant vertical line that

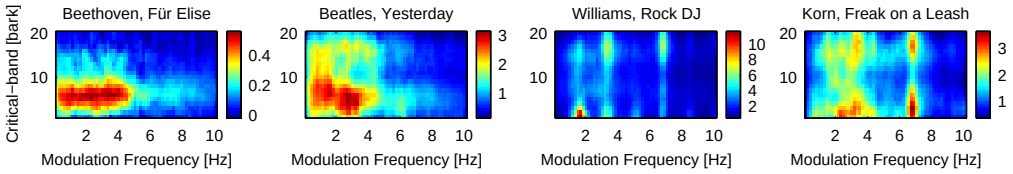


Figure 4.12: The medians of the 4 pieces of music calculated from their sequences (cf. Figure 4.2). The medians are projected and reconstructed from the 80-dimensional eigenspace.

would indicate a strong reoccurring beat.

Some of the sequences of *Rock DJ* seem to have very different patterns. However, detailed analysis shows that they all have something in common, and this is revealed by the median. Similar characteristics apply to *Freak on the Leash*.

Although this analysis might suggest that the median is a good solution this is only partially the case. Many pieces of music might be confined to one genre, even if this genre might be between two other genres, however, especially modern music tends to combine different MFS patterns. On the other hand the dynamic usually does not completely change.

Another advantage of using the median is that the random initializations (GMM-EM, FCM, k-means) are no longer needed and thus the results are very stable and different runs will yield the same results, unlike the previous approaches.

The results using the median can be seen in Figure 4.13. Although this approach is much simpler than the others and can be computed significantly faster, the quality is comparable.

The map unit in the lower left, which represents *elise* (*Für Elise* by *Beethoven*) contains similar classical music such as *air* (*Air from the orchestrasuite #3* by *Bach*), *adagio* (*Adagio from a clarinet concert* by *Mozart*), *kidscene* (*Fremde Länder und Menschen* by *Schuhmann*), and *mond* (*Mondscheinsonate* by *Beethoven*). It is interesting to see that just as with the k-means approach the *mountainking* (*In the hall of the mountainking* by *Grieg*) is mapped together with *elise*. The unit is surrounded by classical music such as *merry* (*The Merry Peasant* by *Schuhmann*), *jurassicpark* (*Jurrasic Park* by *Chuck Berry*), *branden* (*Brandenburgisches Konzert #2* by *Bach*), *forelle* (*Trout - Quintet - Themes and Variations* by *Schubert*), and *walzer* (*Walzer op. 39 No. 15* by *Brahms*).

It is interesting to compare the differences between these units looking at Figure 4.14. The unit representing *elise* (14,1) has the lowest MFS values of all map units. The unit above (13,1) has double as high values and the unit to its right (14,2) has higher activities in the upper critical-bands.

The unit in the center left area of the map which represents *korn-freak* (*Freak on a Leash* by *Korn*) also represents *fbs-praise* (*Praise you* by *Fat Boy Slim*), *limp-99* (*9teen 90nine* by *Limp Bizkit*), and *rem-endoftheworld* (*Its the end of the world* by *REM*). The songs which have been mapped together with *korn-freak* such as *limp-nobodyloves* (*No*

aufderhoeh goldeneye	limp-indigo	macarena	fbs-brighton missingyou	bfmc-fashion bfmc-others rockdj	cocojambo limp-n2gether	seeyouwhen singalongsong	bfmc-1234 kiss rhcp-dirt rhcp-getontop	bongobong themangotree	bfmc-instereo bfmc-rocking bfmc-skylimit bfmc-uprocking letsgetloud wonderland
ga-mine limp-nosex tom	limp-broke	limp-counterfeit limp-faith limp-nookie limp-stuck	limp-justlikethis limp-willbeok mmmbop	ratm-sun rhcp-world	sl-whatigot	bfmc-freestyler sl-summertime	bfmc-rock		aroundtheworld bfmc-flygirls bfmc-stirupthebass latinolover
breathaway ididitagain ifyoubelieve imf	limp-trust pr-infest	limp-method pr-lastresort	limp-finger	badboy mambofive myloveyourlove	tellhim	rhcp-californication rhcp-emitremmus rhcp-scartissue rhcp-universe- rhcp-velvet	rhcp-easily rhcp-otherside saymyname sexbomb	conga	believe eifel65-blue togetheragain
anywhereis lovsisintheair rockisdead	limp-nobody pr-broken pr-revenge	limp-clunk limp-pollution limp-sour pr-snakes	pr-blood	alltoyou johnnyb madlydeeply	ga-gogan jailhouserock	fbs-rockafella fbs-soulsurfing	itsinhiskiss	dschinghiskhan	fbs-island fbs-kalifornia
dayinparadise deepisyourlove lastchristmas	d3-kryptonite limp-rearranged limp-show pr-deadcell wildwildwest	br-punkrock limp-stalemate	verve-bittersweet	aintnosunshine goodgolly	ga-close rhcp-savior running	boogiewoogie feliznavidad	nahnehnah shoopshoopsong	readmymindstars	babycomback duhast movingonup
donttalkanymore nma-125mph piubellacosa rem-beyond	fbs-praise kom-freak limp-99 rem-endoftheworld	limp-lesson	ga-iwantit ga-moneymilk party	br-fiction icouldfly laymedown	grossvater	cabaret diamonds help	walkininmemphis	dingdong everlastinglove ga-doedel surfusa	maniomonday nma-bigblue supertrouper
br-generator nma-lovesongs nma-war sing	limp-nobodyloves pr-neverenough	ga-innocent ga-time	nma-poison whenyousay		joeschau veronika	rhcp-road submarines summerdreaming	br-computer ga-japan	fuerstenfeld	mindfiels panoptikum
africa br-you drive vivaforever	ga-anneclaire ga-heaven limp-wandering pr-binge	addict ga-lie pinkpanther vm-classicalgas vm-toccata	backforgood bigworld br-anesthesia youlearn	allforlove	d3-loser d3-needyou griechischwein lastdance lemontree	cheektocheek	breakfree ga-nospeech matilda	girls	fromnewyorktola gowest radio
dancingqueen philadelphia verve-butterfly vm-red	firsttime holdon	br-jesus		kaktus peggysue	lovedwoman she wonderfulworld	goodbye	lovesombodysomt sanfrancisco unbreakmyheart	allymcbeal zillertaler	breathless duellingviolins poweroflove rem-orange
foreveryoung frozen geldumagstmi sunshineoflife vm-4seasons	ironic	bahnfrei mcgee nma-51st	br-skyscraper verve-luckyman	drummerboy takefive	friend herzilein rhcp-porcelain sml-adia yesterday-b	newyork	icanfly ifonly missathing timewarp		alice br-infected fbs-nighthere rem-religion
blueberry nma-allofthis rocknrollkids	lorddance	nma-dream tritschtratsch	bundeshymne riverdance	goodmornblues	feeling sml-remember	angels anything	americanpie ihavenothing walkingincohn whatsup	bebopalula kisstromarose sport	moonlightshadow nma-race timeaftertime
	beautyandbeast thecircleoflife	allegromolto pr-angels requiem shakespeare	morningbroken nocturne onlyyou starwars threetimesalady yesterday	sml-angel	austria verve-drugs whatsawoman	crashboombang	readmymindlight	nma-questions	br-slumber missyoucrazy
jurassicpark leavingport merry	elvira pachelbl schwan stormsinafrica tell	beethoven lovemetender sml-icecream	adiemus future indy schneib	myway whitechristmas	distance myheartwillgoon	nma-better talkaboutlove	fbs-acid heavensdoor	fatherandson	roadtohell
adagio, air avemaria elise, flute fortuna, funeral kidscene, mond mountainking nachtmusik	branden forelle minuet schindler walzer zapfenstreich zarathustra	fuguedminor therose vm-bach vm-brahms	feellovetonight giubba	donau memory onedaymore rainbow	nma-greengrey radetzkymarsch rem-stand vm-tequila	life phantomopera yougotit	eternalflame nma-brave revolution tannenbaum	rhcp-angels risingsun	californiadream rem-superman summercity tiffany

Figure 4.13: The SOM trained with the pieces of music represented based on the median approach. The blue highlighted map units are the units which represent the 4 pieces of music that are analyzed in detail. The labels are identifiers to which the full title and interpret or author can be found in Appendix B.

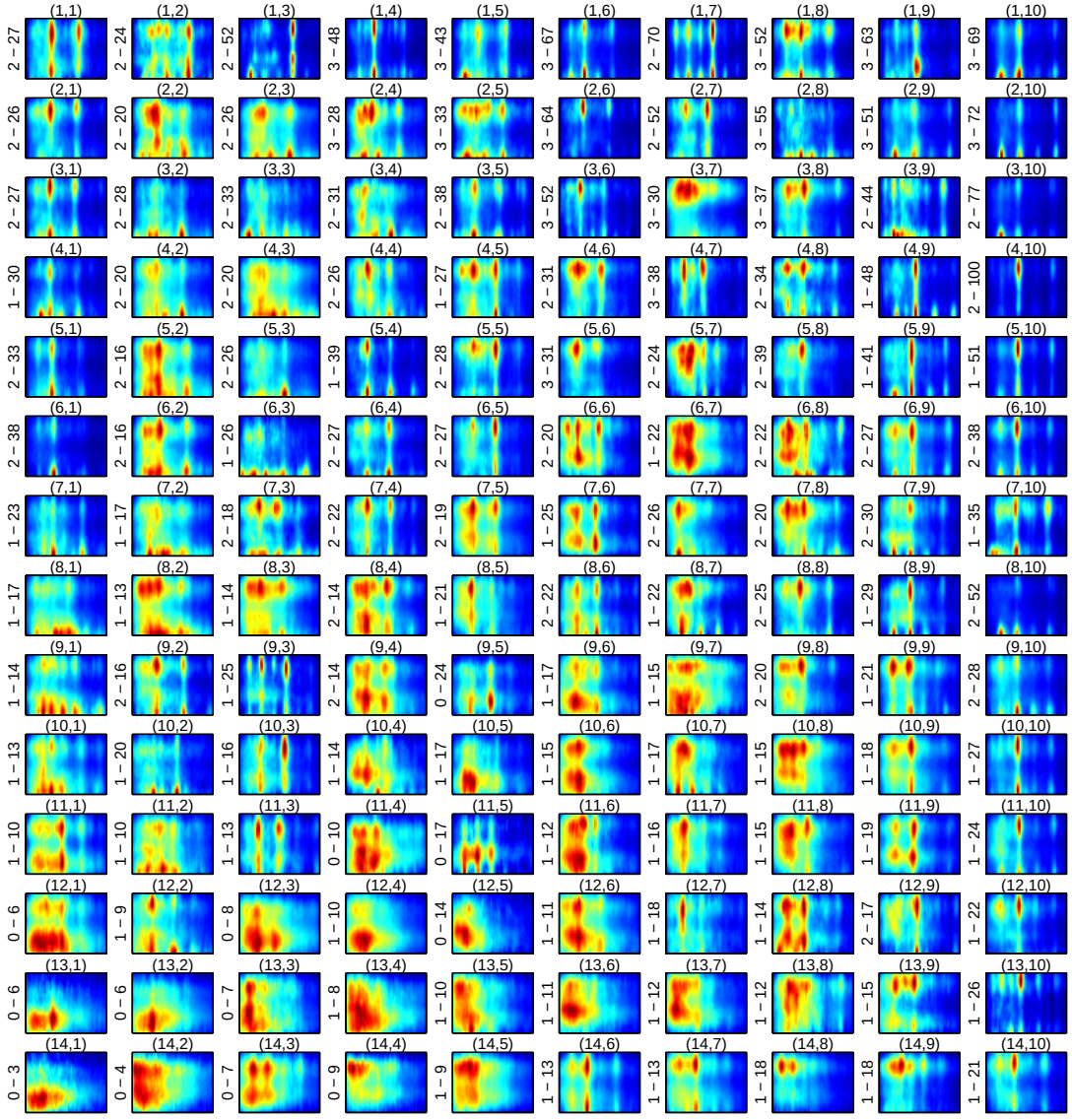


Figure 4.14: The model vectors of the SOM in Figure 4.13. The color coding is scaled to the minimum and maximum of each model vector, the values to the left of each image indicate the relative range of the values, with the highest value in the collection set to 64 and the lowest to 0.

body loves me by *Limp Bizkit*) or *pr-revenge* (*Revenge* by *Papa Roaches*) are mapped to the neighboring units.

The model vector (cf. Figure 4.14) of the unit representing *korn-freak* (6,2) does have some of the characteristics of *korn-freak*, however, the vertical line at the high modulation frequency is not as strong while the fuzzy activiations in the lower modulation frequencies are too strong.

The unit located at the upper center and representing *rockdj* (*Rock DJ* by *Robbie Williams*) also represents the songs *bfmc-fashion* and *bmfc-others* (*Fashion Styley* and *Other EMCEEs* by *Boomfunk MCs*). In the neighborhood are popular songs like *macarena* (*Macarena* by *Los Del Rio*), *mambofive* (*Mambo No 5* by *Lou Bega*), and

cocojambo (*Coco Jambo* by *Mr. President*).

The model vector of the unit (1,5) to which *rockdj* is mapped (cf. Figure 4.14) does not perfectly reflect the properties of the median of *rockdj* (cf. Figure 4.12). However, the unit represents the main characteristics which are the strong bass below 2Hz and the double as fast beats above 3Hz and around 7Hz, which both use all frequencies.

The unit located around the center of the map and which represents *yesterday-b* (*Yesterday* by the *Beatles*) also represents songs like *friend* (*You've got a friend* by *Carole King*), *herzilein* (*Herzilein* by *Wildecker Herzbuam*), *rhcp-porcelain* (*Porcelain* by *Red Hot Chilli Peppers*), and *sml-adia* (*Adia* by *Sarah McLachlan*). Which are all rather slow songs.

The model vector of this unit (10,6) shows no significant vertical lines, instead there are fuzzy activities in the lower modulation ranges in all critical-bands.

The model vectors of the map shown in Figure 4.13 can be seen in Figure 4.14. Generally, the music with lower fluctuation strength is in the lower left and the highest values are around the upper right. Strong beat patterns can be seen especially in the upper left, around the upper right including the area to the lower right.

Note that is also possible to obtain a representation of the MFS for each model vector also with the other presented methods. When using GMM-EM and FCM the model vectors are linear combinations of MFS vectors and thus can be reconstructed easily. When using the k-means approach with the ranking it is not possible to reconstruct exact MFS representations from the model vectors, but it is possible to calculate the MFS of a model vector from the mean of the MFS vectors of the pieces of music mapped to it. The MFS of model vectors to which no pieces of music are mapped can be interpolated from surrounding map units.

4.4. Summary

In this chapter the features extracted in Chapter 3 have been evaluated using the SOM and proven to be a suitable to represent the perceived similarities. Furthermore, different methods to represent the pieces of music based on their sequences have been presented and their performance on the music collection evaluated. Starting with the relatively complex GMM-EM where numerical problems constrained the dimensions of the eigenspace and the number of clusters, which represent the sequences lead to promising yet improvable results. Addressing these problems FCM was used and led to improved results. Alternatively, a heuristically motivated approach based on k-means was evaluated, with comparable results. The good results support the intuitive assumption that one song does not only have one typical beat pattern and instead is a mixture of several different MFS patterns. However, using the median has lead to surprisingly good results as well, furthermore, it is by far the simplest method, hence the median will be used as a basis for the next chapter.

5. Visualization and User Interface

In the previous Chapter SOMs labeled with the song identifiers were used to evaluate the music collection. Basically these maps could be used as user interfaces. Similar songs are located close together on the map and are identified by a string short enough to fit the width of a map unit. While these maps surely are a bigger help then a simple alphabetical list there are two major deficiencies.

The first is the lacking support trying to understand the cluster structure. Looking, for example, at Figure 4.13 it is rather difficult to recognize clusters on the first sight. Only after carefully studying the whole map it appears, for example, that there is a cluster of classical music in the lower left corner. Thus some visual support to identify clusters would be desirable.

The second deficiency derives from the assumption that the music collection and the contained pieces of music are unknown to the user, thus any information on artists or titles are not very useful for the user. To a user a map labeled with unknown music titles, interprets or authors might not be much more useful than randomly generated text. By far more interesting for the user would be some text, which explains what type of music is mapped to a map unit.

In the following sections methods are described which aim at creating a more intuitive user interface. In Section 5.1 the problem of visualizing cluster structure is addressed followed by Section 5.2 where important map areas are summarized and labeled. Both sections use the map presented in Figure 4.13 to illustrate the different possibilities. To enable the reader to explore Islands of Music a small demonstration has been made available on the internet and is briefly presented in Section 5.3. The chapter is summarized in Section 5.4.

5.1. Islands

A useful visualization system should offer the user a good summary of the relevant information. To be effective the distinguishing features (e.g. position, form, and color) in the visual dimensions should be detectable effortlessly and quickly by the human visual system in the preattentive processing phase [Hea96]. In general the efficiency of a visualization will depend on the domain, culture, and personal preferences of the users. The results of the previous chapters can be visualized in several different manners, only a few will be discussed in this section.

This thesis has been titled *Islands of Music* because the metaphor of islands is used to visualize music collections. The metaphor is based on islands, which represent groups of similar data items (pieces of music). These islands are surrounded by the sea, which corresponds to areas on the map where mainly outliers or data items, which do not belong to specific clusters, can be found. The islands can have arbitrary shapes, and

there might be land passages between islands, which indicate that there are connections between the clusters. Within an island there might be sub-clusters. Mountains and hills represent these. A mountain peek corresponds to the center of a cluster and an island might have several mountains or hills.

More accurately a simple approximation of the probability density function is visualized using a color scale which ranges from dark blue (deep sea) to light blue (shallow water) to yellow (beach) to dark green (forest) to light green (hills) to gray (rocks) and finally white (snow). The exact color scale represented by HSV (see e.g. [HB97]) values can be found in Appendix A.2.3. The sea level has been set to 1/3 of the total color range, thus map units with a probability of less then 1/3 are under water. The sea level could be adjusted by the user in real time, the involved calculations are neglectable. The visible effects might aid understanding the islands and the corresponding clusters better.

The density function is estimated using the technique presented in the context of the k-means algorithm (Section 4.3.4). Each piece of music votes for the clusters (map units), which represents it best. The first closest model vector of a corresponding unit gets n points, the second $n-1$, the third $n-2$, and so forth. Thus units, which are close to several pieces of music, will get many points. While clusters, which are not close to any pieces, will hardly have any points. A map unit which is close to many pieces of music is very likely to be close to the center of a cluster, whereas a unit, which does not represent any pieces of music well, is likely to be an intermediate unit between clusters.

In Section 4.3.4 $n = 3$ was used. Thus the closest cluster got 3 points, the second 2, and the third 1. The results using $n = 3$ on the map presented in Figure 4.13 can be seen in Figure 5.1. Notice how clusters such as the classical music in the lower left corner or, for example, the rather isolated island in the upper right corner on which mainly songs by *Bomfunk MCs* are located, immediately become apparent.

The influence of the parameter n can be seen in Figure 5.2. Since this parameter is applied at the very end of the whole process and can quickly be calculated the user can adjust this value in real time when exploring the map, and gain information not only from the structure of islands at a specific parameter setting, but also through analyzing differences between different settings.

The smallest possible value for n is one. When only the closest map unit gets a point this method corresponds to a crisp hit response, which is commonly used in SOM visualizations, for example, in the SOM Toolbox the corresponding function is `som_hits`. Notice how the islands seem very isolated and scattered on the map. The most significant clusters are visible using $n = 1$, for example, the classical cluster on the lower left, or the *Bomfunk MCs* at the upper right. Also other dominant clusters are visible, for example, the island slightly to the lower left of the classical *Bomfunk MCs* islands represents mainly songs by *Red Hot Chili Peppers*.

Usually one could expect that the second best matching unit is located right next to the first best matching. Thus dividing the hit response of a data item 2/3 for the

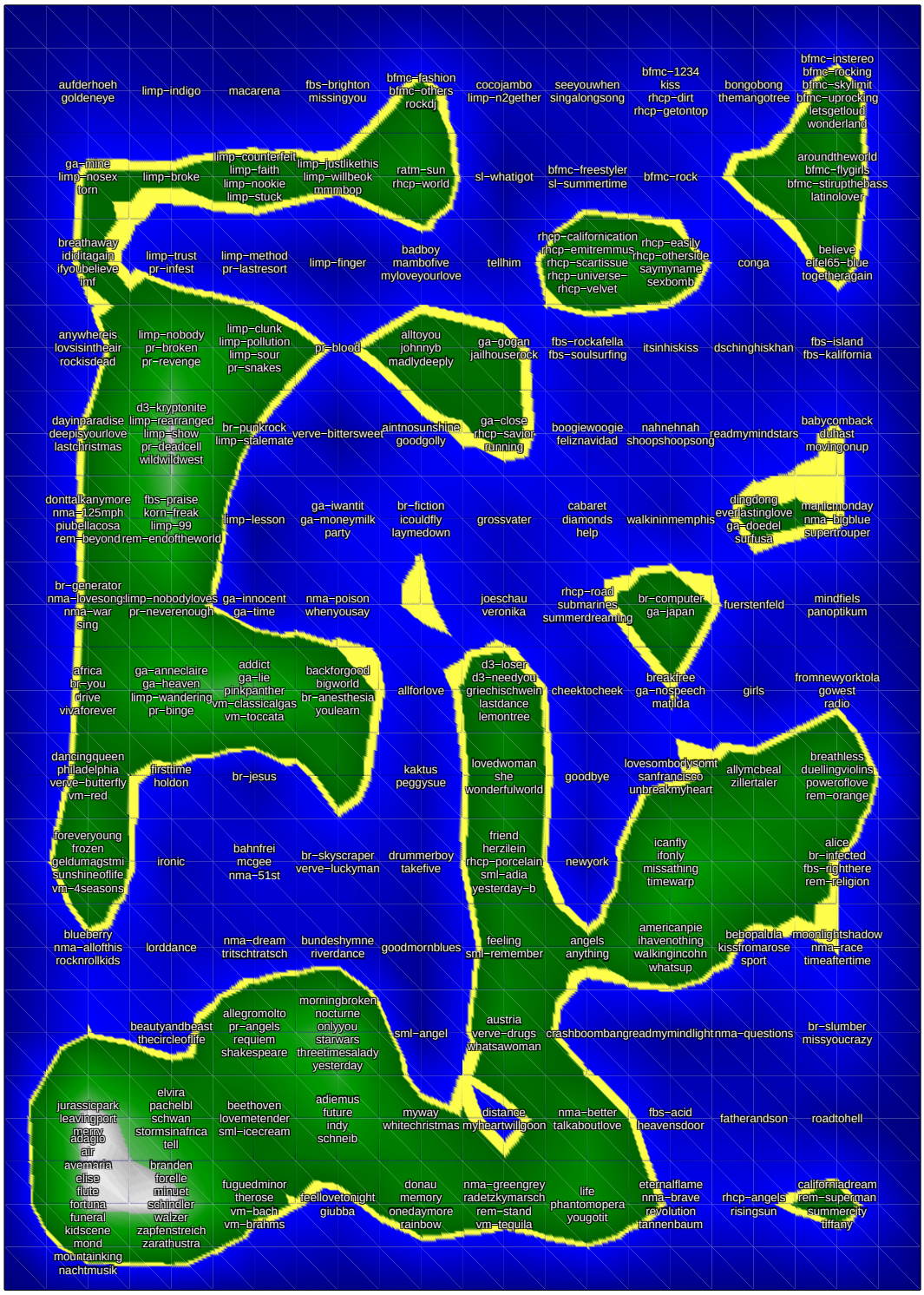


Figure 5.1: The pieces of music ploted on top of the islands visualization.

best matching and 1/3 for the second best matching would lead to the same results with some fuzziness. The often non-spherical shape of the clusters (islands) becomes more apparent and seemingly separated clusters might get connected. For example, the cluster of the classical music at the lower left corner becomes connected with the cluster slightly to the upper right when comparing $n = 1$ with $n = 2$ Figure 5.2. Together these two clusters form a bigger cluster which contains mostly typical classical music, where the slower and more quite pieces are located around the mountain on the lower left of the island and the more powerful classical music is located towards the upper right of the island.

Note that it is not always the case that the first and second best matching units lay next to each other. In fact some quality measures to compare trained SOMs have been developed upon this criteria [KL96, Kiv96]. However, it is rather unlikely that the two units are separated completely on the map and mostly they will both be located in the same map area.

Using $n = 3$ connects more islands and lets them grow bigger. For $n = 4$ the differences to $n = 3$ are not very obvious, however, the islands are slightly more connected and the higher n gets, the more islands will become connected until finally only one big island remains with its peak around the center of the map.

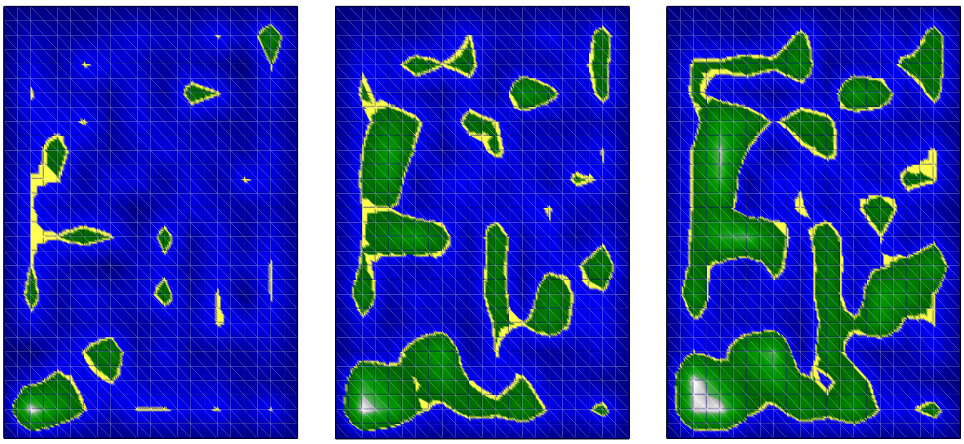
This visualization has been implemented using interpolating units. Between each row and column units have been inserted and around the whole map as well. These interpolating units do not have a corresponding model vector and are only assigned interpolated values of the approximated density function. The units inserted around the map are inserted so that the islands are bounded by the map area. The values of these boarder units are set to 1/10 of their immediate neighbor within the map, and thus are always under water. All density and interpolated density values are then interpolated by Matlab® using the `pcolor` function in combination with `shading interp` to create the islands of Figure 5.2. If desired it would be possible to use fractal algorithms or textures to create more naturally looking islands.

5.1.1. Alternatives

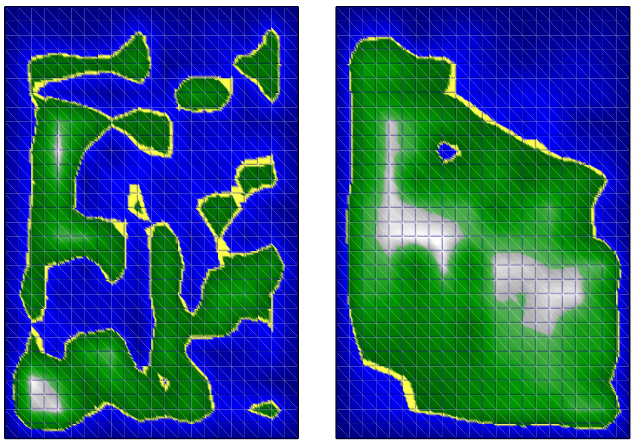
There are several alternatives to the previously proposed method of visualizing islands. The alternatives can be divided into two main categories. There are alternative ways of calculating the estimation of the probability density function, which is the fundament of the islands, and there are alternative methods of visualizing the clusters of a SOM.

Probability Density Function

There are different techniques to calculate an estimation of the probability density function. The most obvious alternative is the Generative Topographic Mapping (GTM) [BSW96]. The GTM is similar to the SOM regarding the results which can be obtained using it, however, the GTM is embedded in a statistical framework and uses



(a) $n = 1$ (b) $n = 2$ (c) $n = 3$



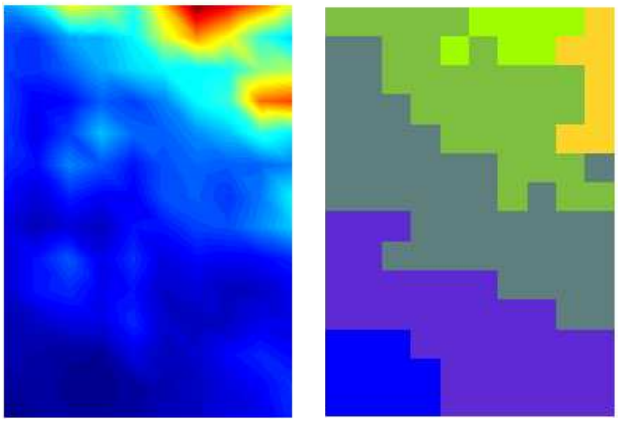
(d) $n = 4$ (e) $n = 100$

Figure 5.2: Islands of Music applied to the map presented in Figure 4.13 with different parameters.

a constrained Gaussian mixture model and expectation maximization to estimate the probability distribution of the data in the 2-dimensional latent space. Similarly the S-Map algorithm [KO97], which combines the softmax activations of the GTM and the learning algorithm of the SOM, could be used.

However, the SOM was chosen because it has proven itself to be a very robust algorithm that scales well with large amounts of data in numerous applications [Koh01]. Since the aim of this thesis is to develop a system which can handle huge music collections the SOM was chosen, which is computationally much lighter then the GTM or S-Map algorithm and less sensitive to the initialization. Since the quality of the results that can be obtained by the SOM, GTM and S-Map have shown to be comparable (see e.g. [RPP00, VK01]) the SOM has been selected.

Once the SOM is trained there are different possibilities to obtain an estimation of the probability density function. Alternatives include methods such as reduced kernel density estimators [Häm95] or much simpler approaches based on the inverse squared



(a) The interpolated median values of the distance matrix.

(b) Color coded k-means with 6 clusters.

Figure 5.3: Alternative visualizations of the SOM presented in Figure 4.13. These figures were created using the SOM Toolbox.

distances of all data items to all units [AHV99]. Recently another approach to calculate the probability density from a given SOM was presented [KL01] based on the local error function of the SOM [RS88] and using Monte Carlo sampling. However, for large maps the computational cost by far exceeds that of the training algorithm itself. However, the proposed ranking method is intuitive, fast to compute and yields good results and thus was preferred.

Visualizations

There are several alternatives to visualize the cluster structure of a SOM. The most common ones are based on the distance matrix, which holds the distances between neighboring units. A well-known representative of these methods is the U-matrix [US90], which visualizes all the distances between the model vectors of adjacent units. These distances can be visualized color coded as proposed in [US90] or, for example, with connecting lines between the units [MR97]. Similar methods include, for example, visualizing only the median of all distances between a unit and its neighbors [KMJ95]. The median of the distance matrix of the SOM presented in Figure 4.13 can be seen in Figure 5.3(a). Notice how it is hardly possible to recognize the cluster structure. Figure 4.14 indicates why the commonly used distance matrix fails to reveal the cluster structure. The distances between the model vectors around the lower left are small because the range of values is small. For example, the values of the unit (14,1) are in the range of about 0-3 and the neighbors have very similar ranges. On the other hand the distances at the upper right of the map are huge because the range of the values is much bigger. For example, the range of the values of unit (4,10) is 0-100 and its neighbors are relatively similar.

Other possibilities include clustering of the SOM using algorithms such as k-means

or the hierarchical single-linkage [VA00]. Figure 5.3(b) depicts the results of clustering the model vectors of the map presented in Figure 4.13 with the k-means algorithm. Each cluster is assigned a color that is defined by the clusters centroid and the location of the corresponding best matching unit on the map. Each unit on the map is assigned a color so that neighboring units have similar colors [KVK99]. Figure 5.3(b) nicely reveals the properties of the model vectors. Especially, the almost linear tendency of the increasing range of the maximum values of the model vectors from the lower left to the upper right is clearly reflected in the k-means clusters.

5.2. Labeling

The islands themselves might not be a very satisfying user interface. Although clusters can be identified, without further information it is not possible to understand what determines the cluster. It is desirable to have a summary for each cluster describing its main characteristics and thus explaining why the map is structured the way it is.

The island visualization can easily be combined with any labeling method. For example, the islands can be labeled with the song identifiers (cf. Figure 5.1). On the other hand these identifiers, when assuming that the songs are unknown to the user, do not help the user understand the clusters.

There are several methods of labeling SOMs such as the LabelSOM technique [Rau99] or the technique used for the WebSOM project [LK99]. Both techniques have been developed especially for the domain of text archives, however, they can be applied in a more general context. The LabelSOM technique uses the mean value and the variance of an attribute within a cluster to decide if it is a good description or not. The WebSOM technique compares the mean values of the attributes within a cluster with those of other clusters and finds descriptors, which characterize some outstanding property of a cluster in relation to the rest of the collection. Other techniques include inductive machine learning algorithms, which extract fuzzy rules to describe the clusters [Ult91, UK95, DWB00].

All these techniques have in common that they find descriptions based on the dimensions and their meanings. For example, in the domain of text archive analysis it is common to use the vector space model [SM83], where the text documents are represented in a high-dimensional space, with each dimension assigned to some word. The labeling techniques try to find the most descriptive dimensions (i.e. words) and use these to describe the map.

This cannot be applied directly to the music collection and the methods introduced in this thesis since the single dimensions of the 1200-dimensional modified fluctuation strength vectors are rather meaningless. For example, the MFS value at the modulation of 2.4Hz and bark 4 would not be very useful describing a specific type of music.

5.2.1. Aggregated Attributes

Since the single dimensions are not very informative aggregated attributes can be formed from these and can be used to summarize characteristics of the clusters. There are several possibilities to form aggregated attributes, for example, using the sum, mean, or median of all or only a subset of the dimensions. Furthermore, it is possible to compare different subsets to each other. In the following 4 aggregated attributes, which point out some of the possibilities, are presented. If the user understands the MFS it is possible that the user directly creates the aggregated attributes depending on personal preferences.

The presented aggregated attributes are *Maximum Fluctuation Strength*, *Bass*, *Non-Aggressive*, and *Low Frequencies Dominant*. The names have been chosen to indicate what they describe.

The *Maximum Fluctuation Strength* (cf. Figure 5.4(a)) is defined as the highest value in the MFS. Pieces of music, which are dominated by strong beats, usually have very high values. Whereas, for example, classical piano pieces like *Für Elise* have very low values.

The *Bass* (cf. Figure 5.4(b)) is calculated as the sum of the MFS values in the lowest two critical-bands (bark 1 and 2) and a modulation frequency higher then 1Hz. Thus describes the energy of the bass beats with at least 60bpm.

To compute the *Non-Aggressive* attribute (cf. Figure 5.4(c)) each of the MFS model vectors is normalized by its maximum value so that its highest value equals 1. This corresponds to what can be seen in Figure 4.14 where each model vectors is scaled to make best use of the color scale. From the normalized MFS the sum of the values in the critical-bands from bark 3 to bark 20 with a modulation frequency below 0.5Hz (30bpm) is calculated. From Figure 4.14 it can be seen that especially model vectors, which do not have strong vertical lines, have high relative values in that range.

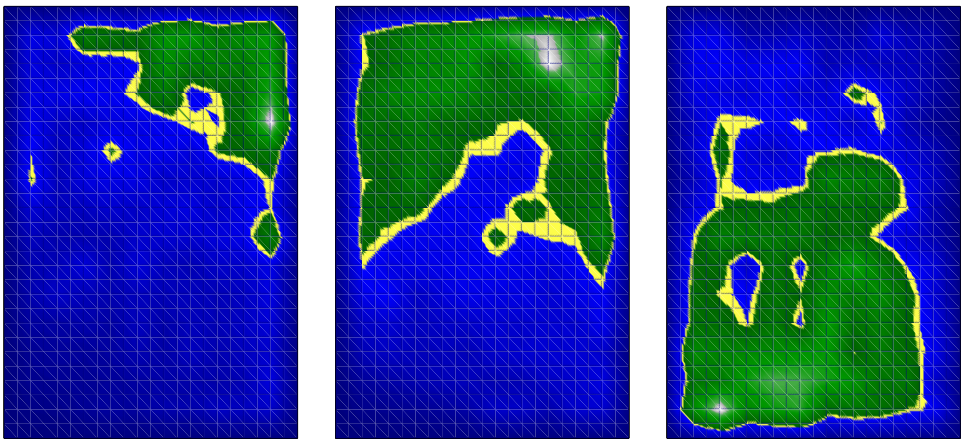
The *Low Frequencies Dominant* attribute (cf. Figure 5.4(d)) is calculated as the ratio between the energy contained in the critical-bands from bark 1 to 5 and the critical-bands from bark 15 to 20. This attribute indicates if a piece of music has most of the MFS energy in the lower or in the upper frequency spectrum.

Figure 5.4 shows the so-called *component planes*, which are visualized in the same way as the islands of music. Mountains represent high values, water represents very low values, of the corresponding aggregated attribute (i.e. component).

It is easily possible to construct arbitrary other aggregated attributes and visualize them in the same way. The component plane visualization can also be used to analyze the correlation between different attributes [VA99].

5.2.2. Rhythm

Other interesting aspects of the MFS include the vertical lines that correspond to beats at specific frequencies. To extract information on these a similar method as used previ-



(a) Maximum Fluctuation Strength

(b) Bass

(c) Non-Aggressive

(d) Low Frequencies Dominant

Figure 5.4: The component planes of the aggreagated attributes.

ously (page 31) can be used to find significant beats. First the sum over all critical-bands for each modulation frequency of the MFS are calculated and normalized by the maximum value so that the highest value equals one. The results for the map presented in Figure 4.13 can be seen in Figure 5.5. High peeks correspond to significant beats and are found using the following rules. (1) All peeks below 43% of the maximum are ignored. (2) All other peeks need to be at least 12% higher then the closest preceding and the closest succeeding minima. (3) Finally from the remaining peeks only those are selected which are higher then any other remaining peek within the modulation frequency range of 1Hz.

5.2.3. Results

For each label a summary consisting of aggregated attributes and information on the rhythm can be given. These descriptions could be placed directly on the map of is-

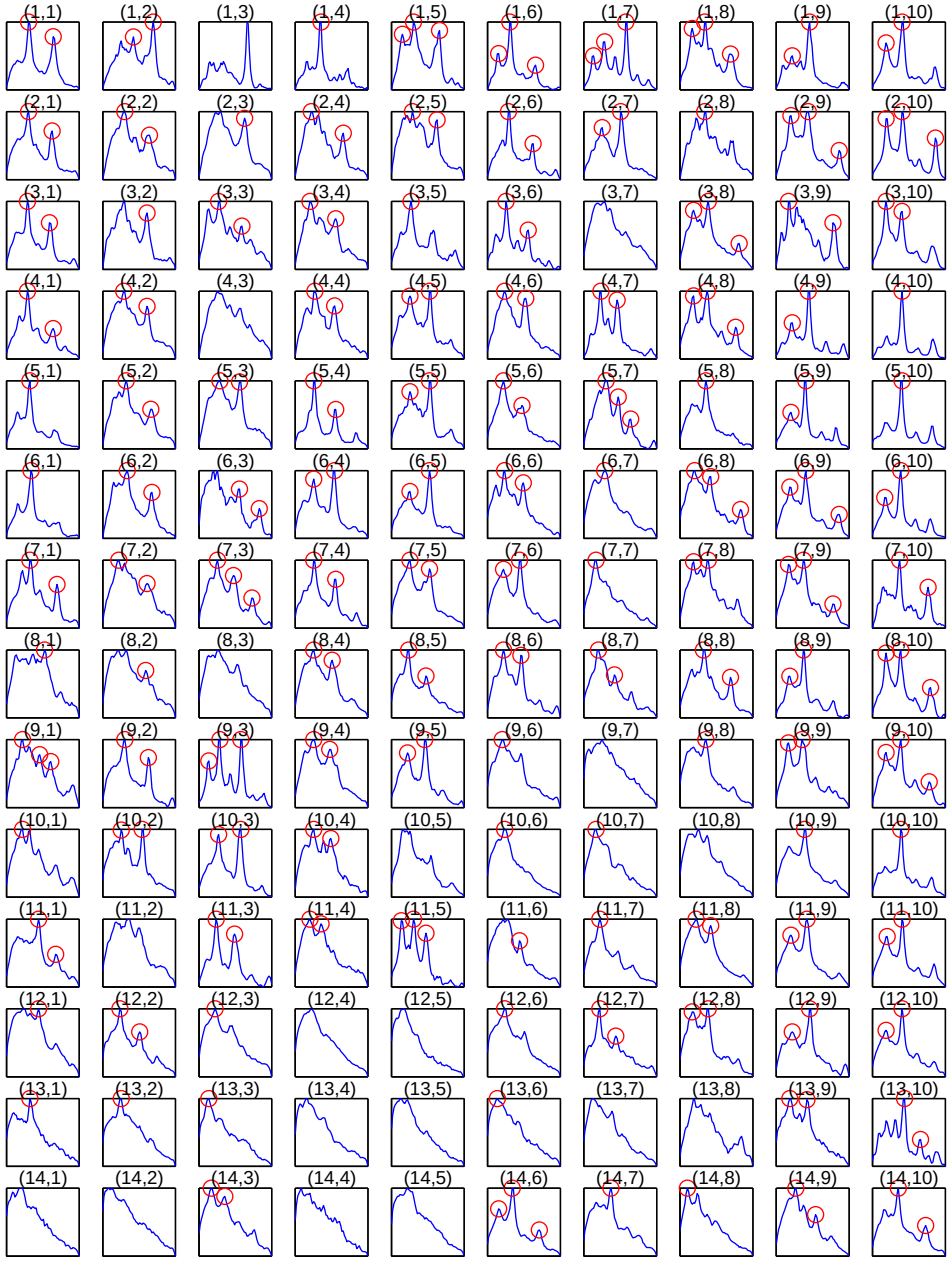


Figure 5.5: The sum of the MFS in the critical-bands versus the modulation frequency.

lands, however, describing each map unit would lead to 140 (10x14) descriptions and thus would need to be summarized again. Instead it would be desirable to only label significant areas on the map, for example, mountains or hills. In Figure 5.6 all peeks have been labeled which are above the sea level.

The exact values of the component planes might not be very interesting for the user. Furthermore, it might only be interesting to summarize unusual characteristics of a mountain compared to the rest of the collection. Thus instead of using the exact values these were coded using ++, +, -, and --. The ++ is used only if the value is at least 7/8 of the highest value in the collection, + only if the value is at least 5/8 of the

highest, - for not more than 3/8, and -- for values below 1/8. Values between 3/8 and 5/8 are ignored since they are not considered to be outstanding.

The summary about rhythm is limited to the frequency of the significant beats given in bpm. If the MFS value of a beat is 50% of the highest a "!" is added to emphasize it.

The automatically generated summaries of the characteristic map areas are not comparable to what a human could produce, however, they help understand the structure of the map. For example, the classical music cluster in the lower left of the map is labeled with *maxflux–, bass–, low-freq-dom++,192bpm* slightly above the lower left corner and slightly to the right of the lower left corner are the labels *maxflux–, bass–, non-aggressive++, low-freq-dom-*. From these labels the user can conclude that the music in this area does not contain much bass *(bass–)* and generally has no strong beats *(maxflux–)*. The *low-freq-dom++* label describes the characteristics of piano music. For example, *Für Elise* mostly uses the lower half of the critical-bands. Note that the *192bpm* reflect the quickly but quietly played tones in most of the piano pieces. Obviously this is a deficiency of the technique used to extract this information, which uses relative and not absolute values.

On the other side of the map, the upper right corner, the labels summarize the characteristics of music such as the songs by *Bomfunk MC's*. These songs have strong beats *(maxflux+)* and a strong bass *(bass+)*. They cannot be described as non-aggressive *(non-aggressive–)* and the labels (111bpm!, 242bpm!, 515bpm) indicate that these songs have very fast beats.

5.3. HTML Interface

To enable the reader to explore and to listen to Islands of Music a small demonstration has been made available on the internet[1]. For this demonstration only a subset of 77 songs from the music collection was used and a 7x7 SOM was trained.

The user has the choice to view several images of the maps, the component planes, and details on the model vectors. The images are presented so that the user can easily link these and thus can easily understand the multi-dimensional information [BMMS91].

The following maps can be viewed. (1) The basic 7x7 SOM with the units labeled with the song identifiers, (2) the map of islands with the units labeled with song identifiers, (3) the plain map of islands, (4) the map of islands with white balls representing the pieces of music, and (5) the map of islands where the mountains and hills are labeled as presented in the previous section.

The map with white balls is a HTML *image map* where the white balls are linked to the respective MP3 files. Furthermore, holding the mouse over one of the balls will display the tooltip[2], which contains information on author, interpret, and title. The

[1] `http://student.ifs.tuwien.ac.at/~elias/music`

[2] Tested with Internet Explorer 5.5 and Netscape Navigator 4.08.

Figure 5.6: The SOM presented in Figure 4.13 using the island visualization with voting parameter 3 and labels which describe the clusters based on the rhythmic and other aggregated attributes.

white balls are placed so that they do not overlap each other, this is implemented by randomly placing them using a normal distribution with a small variance, and then rearranging them until there are no more overlaps. Alternatively, Sammon's mapping could be used or a local PCA could be implemented similar to technique presented by Kaski in the context of analyzing the SOM cluster structure [KNK98]. This map is the main interface to the unknown music collection, which the user intends to explore, with the possibility to directly listen to the songs on the map. It would be desirable to add features like the possibility to mark certain map areas or pieces of music, and this might be implemented in the near future. All maps can be viewed with the voting parameter set to 1, 2, or 3. A better interface would be a sliding bar that covers all possibilities (on a 7x7 map the highest possible voting parameter is 49).

The images of the component planes are the same size as the islands of music images to enable the user to link them more easily. All 4 presented aggregated attributes can be viewed.

The information on the model vectors of the map units are intended for more advanced users with basic understandings of the underlying concept. The scaled, unscaled MFS as well as the MFS rhythms are available.

Figure 5.7 shows a screenshot of the web page where the frame on the left contains direct links to the different visualizations that are displayed in the main frame on the right hand side. The current frame shows the interface with the white balls and the cursor over *Für Elise*.

5.4. Summary

In this chapter a new method to visualize the clusters of the SOM was presented. This method uses islands to symbolize clusters. Within clusters sub-clusters are visualized using mountains or hills depending on their size. Although the proposed method is related to the probability density function it is heuristically motivated and lacks any statistical background. However, the experiments have shown that the method is very robust and outperforms common methods depending on the used dataset.

Furthermore, possibilities to label the map with descriptions, which summarize the properties of the mapped data, were presented. The presented labels are generated directly from the MFS data and point-out a wide range of possibilities.

And finally a HTML demonstration was briefly presented which is available on the internet and implements the user interface described in this thesis.

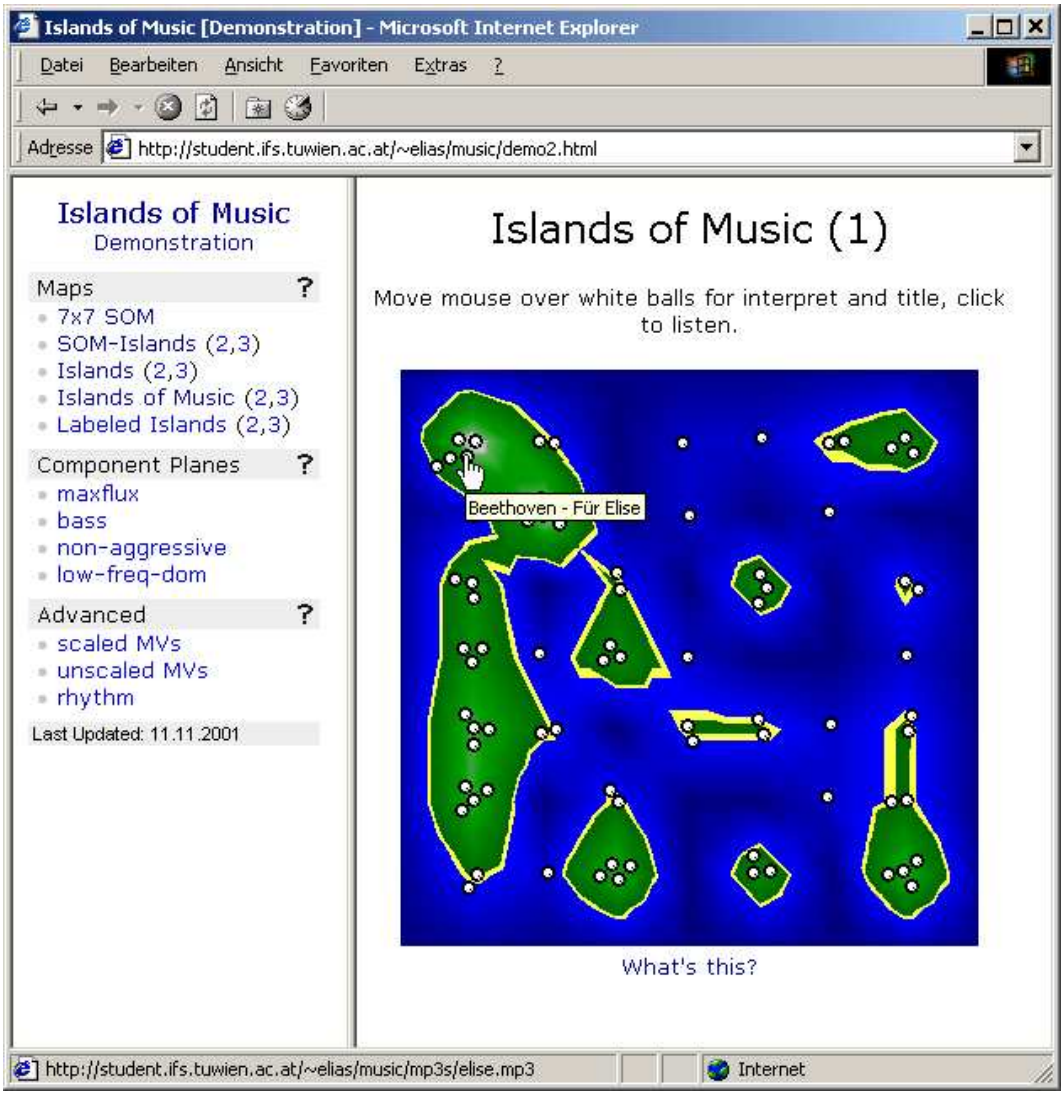


Figure 5.7: Screenshot of the web page implemented to demonstrate the Islands of Music.

6. Conclusion

This final chapter summarizes the work presented in this thesis. In addition, opportunities for future research are pointed out and briefly discussed.

6.1. Summary

In this thesis models and techniques from the fields of signal processing, psychoacoustics, image processing, and data mining were combined to develop a system which automatically builds a graphical user interface to music archives, given only the raw music collection with no further information, such as the genres to which the pieces of music belong.

The most challenging part is to compute the perceived similarity of two pieces of music. Even though currently no final solution to this can be offered, a novel and straightforward approach based on psychoacoustic models is presented and evaluated using a collection of 359 pieces of music. Despite being far from perfect, this approach yields encouraging results.

A neural network algorithm, namely the self-organizing map, is applied to organize the pieces of music so that similar pieces are located close together on a 2-dimensional map display. A novel visualization technique is applied to obtain the map of islands, where the islands represent clusters in the data. To support navigation in unknown music collections, methods to label landmarks, such as mountains or hills, with descriptions of the rhythmic and other properties of the music located in the respective area, are presented.

The Islands of Music have not yet reached a level, which would suggest their commercial usage, however, they demonstrate the possibility of such a system and serve well as a tool for further research. Any part of the system can be modified or replaced and the resulting effects can easily be evaluated using the graphical user interface.

6.2. Future Work

Much research is being conducted in the area of content-based music analysis with new results being published frequently. Incorporating these results into the presented system would increase the quality of the Islands of Music.

Based on the approach presented in this thesis there are some immediate possibilities that might yield improvements. One major problem is the loudness of the pieces of music. Most pieces in the presented collection are from different sources with significant differences regarding the recorded loudness. The loudness has direct impact on the perceived beats and thus strong influence on the whole system. Methods to normalize

the loudness would increase the quality.

Another interesting aspect is the sequencing. Each piece of music is divided into 6-second sequences and only a small subset of these sequences is further analyzed to reduce the computational load. Using more sequences or even overlapping them would result in more accurate representations of the pieces of music.

Furthermore, the optimal length of the sequences is not yet decided. When calculating the amplitude modulation (Section 3.3.1) less then half of the obtained FFT coefficients are used, in particular those which correspond to the modulation frequencies below 10Hz. Thus it would be possible to reduce the length of the sequences to 3 seconds without modifying any other parts of the system. The advantage of a shorter sequence is that it is less likely that it contains more than one specific style.

The applied image processing filters, which emphasize important aspects in the fluctuation strength images, need to be evaluated more thoroughly as well. Alternative parameter settings as well as alternative filters should be considered.

In this thesis several methods to represent a piece of music based on the representation of its sequences were discussed. The finally chosen method, the median, does not coincide with intuitive assumptions, however the alternatives presented were not able to produce significantly better results. A thorough analysis is necessary and perhaps a method, which combines the advantages of the median with the advantages of the other methods presented, could be developed.

Depending on the dataset alternative ways to label the mountains and hills on the islands could be developed which better help in understanding the genres they represent. It is unlikely that such improved labels can be derived directly from the presented modified fluctuation strength (MFS) data thus the incorporation of different content-based approaches to analyze music is desirable.

A. Source Code and Constants

This appendix includes the Matlab® source code and the constant matrixes referenced in previous chapters. For the context and description please see the corresponding chapters.

A.1. Source Code

The internal Matlab® functions are very efficient, while all others, if not compiled, are interpreted by Matlab® during execution time. This significantly slows down especially functions with loops. All source code presented here was optimized in regard to the execution time, thus avoiding loops where possible and instead using built in Matlab® functions.

A.1.1. Batch SOM

This implementation is similar to the `som_batch_train` function in the SOM Toolbox[1] [Ves00a, VHAP00]. However it is a very reduced version which is optimized for high-dimensional data. From the code below, some additional functionality such as visualization, convergence information, and options regarding neighborhood type, map grid type, and initialization have been removed to emphasize the relevant parts.

```
function M = som_training(D,x,y,radius)
% D ....... matrix, dataset, rows are items, columns are dimensions
% x, y .... scalar, map size
% radius .. vector, neighborhood radius (sigma) for each iteration
% M ....... matrix, model vectors

m = x * y;       % number of (map) units
[n d] = size(D); % number of dataitems and their dimension

M = rand(m,d)+repmat(mean(D),m,1)-0.5; % random init with small values

% unit coordinates in rect 2-dim output (latent) space
    Cxy = [reshape(repmat(1:x,y,1),1,m); repmat(y:-1:1,1,x)]';
% Ud(i,j): distance between the units i and j in the latent space
    Ud = zeros(m,m);
    for i = 1:(m-1),
        Ud(i,i+1:m) = sum((Cxy(i+1:m,:)-repmat(Cxy(i,:),m-i,1))'.^2);
    end
    Ud = Ud + Ud'; % symmetric

% optimize training loop
    radius = radius.^2;
    D2=2*D';
```

[1] http://www.cis.hut.fi/projects/somtoolbox

```
for t = 1:length(radius) % training loop
    % quantisation error Q, winner W (unit with smallest distance to data item)
        [Q W] = min(repmat(sum(M.^2,2),1,n)-M*D2);
    % partition P(i,j)=1 if unit i is activated by dataitem j else P(i,j)=0
        P = sparse(W,1:n,ones(n,1),m,n);
    N = exp(-Ud/(2*radius(t))); % gaussian neighborhood function
    S = N*(P*D); % weighted sum of mapped vectors
    A = N*sum(P,2); % activation normalization factor
    % update only units with activation > 0
        nonzero = find(A > 0);
        M(nonzero,:) = S(nonzero,:) ./ repmat(A(nonzero),1,d);
end % training loop
% END of function SOM_TRAINING
```

A.1.2. Fuzzy C-Means

This fuzzy c-means (FCM) implementation has been optimized for high-dimensional data. For simplicity this function implements a random initialization, although the experiments were carried out using a k-means initialization.

Note that it would easily be possible to define a stoping criteria based on the change in the error function, which has been omitted for simplicity.

```
function [C,U] = fcm(D,c,iter,b)
% D ..... matrix, dataset, rows are items, columns are dimensions
% c ..... scalar, number of clusters
% iter .. scalar, number of iterations
% b ..... scalar, FCM parameter
% C ..... matrix, cluster centers
% U ..... matrix, memberships

[vecs dims] = size(D);

% random initialization
U = rand(c, vecs);
U = U./repmat(sum(U),c,1);

% optimize training loop
D2 = 2*D';
const = repmat(sum(D'.^2,1),c,1);

for i=1:iter, % training loop
    U2 = U.^b;
    C = U2*D./repmat(sum(U2,2),1,dims); % new centers
    distance = repmat(sum(C.^2,2),1,vecs) - C*D2 + const;
    U = sqrt(distance).^(2/(1-b));
    U = U./repmat(sum(U),c,1);
end
% END of function FCM
```

A.1.3. K-Means

The k-means algorithm is very similar to the batch SOM. It is not necessary to calculate the exact distances, it is only necessary to know which cluster is clostest to each data item. Unlike the SOM k-means does not train the neighborhood of a cluster thus,

a cluster is only set to the average of the data items closest to it. This Matlab® implementation has been optimized for high dimensional data.

```
function C = kmeans(D,iter,c)
% D ..... matrix, dataset, rows are items, columns are dimensions
% iter .. scalar, number of iterations
% c ..... scalar, number of clusters
% C ..... matrix, cluster centers

[n d] = size(D);

% random initialization
    perm = randperm(n);
    C = D(perm(1:c),:);

D2=2*D'; % optimize training loop

for t = 1:iter % training loop
    % quantisation error Q, winner W (unit with smallest distance to data item)
        [Q W] = min(repmat(sum(C.^2,2),1,n)-C*D2);
    % partition P(i,j)=1 if unit i is activated by dataitem j else P(i,j)=0
        P = sparse(W,1:n,ones(n,1),c,n);
    S = P*D; % sum of mapped vectors
    A = sum(P,2); % activation normalization factor
    % update only units with activation > 0
        nonzero = find(A > 0);
        C(nonzero,:) = S(nonzero,:) ./ repmat(A(nonzero),1,d);
end % training loop
% END of function KMEANS
```

A.2. Constants

The matrixes are in Matlab® syntax. The first index is the row index, the second index is the column index.

A.2.1. Equal Loudness Contours

These values are based on the equal loudness contour diagram in [ZF99]. The matrix contains the decibel values for the i-th contour level for the j-th critical-band in $\mathbf{C}_{elc}(i, j)$. The contour levels, starting at the first level, correspond to 3, 20, 40, 60, 80, and 100 phon.

```
C_elc(:,1:10) = [ ...
   24   14   11    8    6    5    4    3    3    2;
   37   27   24   22   21   20   20   20   20   20;
   51   43   41   41   41   40   40   40   40   40;
   70   63   61   61   60   60   60   60   60   60;
   88   83   81   81   80   80   80   80   80   80;
  105  102  101  101  100  100  100  100  100  100];

C_elc(:,11:20) = [ ...
    2    2    0   -1   -3   -5   -4   -2    1    3;
   20   20   19   17   15   13   13   15   19   21;
   40   39   38   36   34   33   33   34   36   39;
   60   59   56   54   53   52   53   54   57   59;
```

```
  80   79   76   74   71   70   70   72   76   78;
 100   99   97   95   93   90   90   92   96   98];
```

A.2.2. Modified Fluctuation Strength

These matrixes are basically Gaussian filters. Note that edges are not treated in the usual way.

```
function S = spread_matrix(n,w) % n x n matrix, with weights w
S = eye(n);                     % identity matrix
for i = 1:length(w),
    S(1:end-i,1+i:end) = S(1:end-i,1+i:end)+w(i)*eye(n-i);
    S(1+i:end,1:end-i) = S(1+i:end,1:end-i)+w(i)*eye(n-i);
end
S = S./repmat(sum(S,2),1,n);    % normalize to sum = 1
```

Critial-Band Spread

```
n = 20;                  % number of critical-bands
w = [0.5 0.25 0.1 0.05]; % symmetrical weighting factors (excl. 1)
S_i = spread_matrix(n,w);
```

Modulation Frequency Spread

```
n = 60;                  % number of modulation frequencies
w = [0.5 0.25 0.1 0.05]; % symmetrical weighting factors (excl. 1)
S_n = spread_matrix(n,w);
```

A.2.3. Color Scale

These values are the HSV values used to create the islands metapher. The Matlab® `colormap` function expects a matrix consisting of 64 rows with 3 columns, which represents 64 colors and their red, green and blue components.

```
% HUE
h=[...
     ones(1,22)*2/3,...       % water (22)
     60/360,60/360,...        % beach (2)
     ones(1,20)*1/3,...       % forest (20)
     ones(1,12)*1/3,...       % gras (12)
     ones(1,8)]';             % mountain (8)

% SATURATION
s=[...
     ones(1,22),...           % water (22)
     0.7,0.7,...              % beach (2)
     ones(1,20),...           % forest (20)
     linspace(1,0.5,12),...   % gras (12)
     zeros(1,8)]';            % mountain (8)

% VALUE
v=[...
     linspace(0.5,1,22),...   % water (22)
     1,1,...                  % beach (2)
     linspace(0.4,0.6,20),... % forest (20)
```

```
    linspace(0.6,0.7,12),... % gras (12)
    linspace(0.7,1,8)]';     % mountain (8)

colormap(hsv2rgb([h,s,v]))
```

B. Music Collection

In this appendix the pieces of music are listed which have been used in the expriments presented in this thesis. A large part of the collection is the same as used in [Frü01], some pieces of music were added which have been copied directly from music albums. One of the main problems with the collection is the not normalized loudness. A human listener would wish to adjust the volumn when listening to the classical pieces of music in the collection. Some pieces are recorded too quiet, some too loud. The same applies for all other genres, a normalization would be desirable, however loudness normalization is a very complex topic and thus in this thesis no further attempts were made other than normalizing the sound pressure amplitudes or the specific loudness, which both have not lead to the desired effects. Since the loudness has a major impact on the impression of music [ZF99] future experiments should focus on collections with normalized loudness.

Note that the qualility of the music (e.g. sampling rate) does not play a major role for the presented method, as long as the main beat characteristics are not influenced.

The collection includes 359 pieces of music out of several different genres. The total length is 23 hours. The shortest piece is 49 seconds and the longest is about 9.5 minutes long. The average duration is about 3.8 minutes.

Table B.1: Music Collection.

Identifier	Title	Interpret or Author
adagio	Adagio aus Klarinettenkonzert	Mozart
addict	Addict	K's Choice
adiemus	Adiemus	Adiemus
africa	Africa	Toto
aintnosunshine	Ain't no Sunshine	Lighthouse Family
air	Air aus Orchestersuite #3	Bach
alice	Living Next Door To Alice	Smokie
allegromolto	Allegro Molto	Brahms
allforlove	All For Love	Brain Adams, Rod Steward, Sting
alltoyou	All To You	The Rounder Girls
allymcbeal	Searching My Soul	Vonda Shepard
americanpie	American Pie	Don McLean
angels	Angels	Robbie Williams
anything	Anything goes	Tony Bennett
anywhereis	Anywhere Is	Enya
aroundtheworld	Around the world	ATC
aufderhoeh	Auf der Höh	Landler (Ziehharmonika)
austria	I am from Austria	Rainhard Fendrich
avemaria	Ave Maria	Schubert
babycomback	Baby come back	The Equals
backforgood	Back for good	Take That
badboy	Bad Boy	Gloria Estefan
bahnfrei	Bahn frei - Polka schnell	
beautyandbeast	Schöne und das Biest	Titelsong
bebopalula	Bebopalula	Gene Vincent
beethoven	5th Symphony 1st Movement	Beethoven
believe	Believe	Cher
bfmc-1234	1,2,3,4	Bomfunk MCs
bfmc-fashion	Fashion Styley	Bomfunk MCs
bfmc-flygirls	B-Boys and Flygirls	Bomfunk MCs
bfmc-freestyler	Freestyler	Bomfunk MCs
bfmc-instereo	In Stereo	Bomfunk MCs
bfmc-others	Other EMCEEs	Bomfunk MCs
bfmc-rock	Rock, rocking the Spot	Bomfunk MCs
bfmc-rocking	Rocking, just to make ya move	Bomfunk MCs
bfmc-skylimit	Sky's the limit	Bomfunk MCs
bfmc-stirupthebass	Stir up the bass	Bomfunk MCs
bfmc-uprocking	Uprocking Beats	Bomfunk MCs
bigworld	Big Big World	Emilia
blueberry	Blueberry Hill	Fats Domino
bongobong	Bongo Bong	Manu Chao
boogiewoogie	Boogie Woogie Bugle Boy	Bette Midler
br-anesthesia	Anesthesia	Bad Religion
br-computer	I love my computer	Bad Religion
br-fiction	Stranger than fiction	Bad Religion
br-generator	Generator	Bad Religion
br-infected	Infected	Bad Religion
br-jesus	American Jesus	Bad Religion
br-lovesongs	Lovesongs	New Model Army
br-punkrock	Punk Rock Song	Bad Religion
br-skyscraper	Skyscraper	Bad Religion
br-slumber	Slumber	Bad Religion
br-you	You	Bad Religion
branden	Brandenburgische Konzert #2 Andante	Bach
breakfree	I Want To Break Free	Queen
breathaway	Take My Breath Away	Berlin
breathless	Breathless	The Corrs
bundeshymne	Bundeshymne	Mozart
cabaret	Cabaret	Liza Minelli
californiadream	California Dreaming	Mamas and the Papas
cheektocheek	Cheek to cheek	Ella Fitzgerald
cocojambo	Coco Jambo	Mr President
conga	Conga	Gloria Estefan
crashboombang	Crash Boom Bang	Roxette
d3-kryptonite	Kryptonite	3 doors down
d3-loser	Loser	3 doors down
d3-needyou	So I need you	3 doors down
dancingqueen	Dancing Queen	ABBA
dayinparadise	Another Day In Paradise	Phil Collins
deepisyourlove	How Deep Is Your Love	Take That
diamonds	Diamonds are a girls best friend	Marilyn Monroe
dingdong	Ding Dong	EAV
distance	From a Distance	Bette Midler
donau	Donauwalzer	
donttalkanymore	We Don't Talk Anymore	Cliff Richard
drive	Drive	The Cars
drummerboy	Little drummer boy	Bing Crosby, David Bowie
dschinghiskhan	Dschinghis Khan	Dschinghis Khan
duellingviolins	Duelling violins	Feet of Flames
duhast	Du Hast	Rammstein (Matrix)
eifel65-blue	Blue	Eifel 65
elise	Für Elise	Beethoven

Identifier	Title	Interpret or Author
elvira	Andante Klavierkonzert #21 Elvira Madigan	Mozart
eternalflame	Eternal Flame	Bangles
everlastinglove	Everlasting Love	Gloria Estefan
fatherandson	Father And Son	Cat Stevens
fbs-acid	Acid 8000	Fat Boy Slim
fbs-brighton	Youre not from Brighton	Fat Boy Slim
fbs-island	Love islands	Fat Boy Slim
fbs-kalifornia	Kalifornia	Fat Boy Slim
fbs-praise	Praise you	Fat Boy Slim
fbs-righthere	Right here right now	Fat Boy Slim
fbs-rockafella	Rockafella Skank	Fat Boy Slim
fbs-soul-surfing	Soul Surfing	Fat Boy Slim
feeling	You've lost that lovin' feeling	Rightous Brothers
feellovetonight	Can You Feel The Love Tonight	Elton John
feliznavidad	Feliz Navidad	Jose Feliciano
firsttime	The First Time	Robin Beck
flute	Flötenkonzert in G minor - andante	Buffardin
forelle	Trout - Quintet - Themen und Variationen	Schubert
foreveryoung	Forever young	Rod Stewart
fortuna	O Fortuna Imperatrix Mundi	Carl Orff
friend	You've Got A Friend	Carole King
fromnewyorktola	From New York to L.A.	Stephanie McKay
frozen	Frozen	Madonna
fuerstenfeld	Fürstenfeld	STS
fuguedminor	Toccata and Fugue in D Minor	Bach
funeral	Begräbnismarsch (Piano Sonate #2)	Chopin
future	End Credits	Back To The Future II
ga-anneclaire	Anne Claire	Guano Apes
ga-close	Too close to leave	Guano Apes
ga-doedelup	Dödel Up	Guano Apes
ga-gogan	Gogan	Guano Apes
ga-heaven	Heaven	Guano Apes
ga-innocent	Innocent Greed	Guano Apes
ga-iwantit	I want it	Guano Apes
ga-japan	Big in Japan	Guano Apes
ga-lie	Living in a lie	Guano Apes
ga-mine	Mine all mine	Guano Apes
ga-moneymilk	Money and Milk	Guano Apes
ga-nospeech	No Speech	Guano Apes
ga-time	Aint got time	Guano Apes
geldumagstmi	Gel Du Magst Mi	Ludwig Hirsch
girls	Girls Girls Girls	Sailor
giubba	Vesti La Giubba (3 Tenors)	Domingo
goldeneye	GoldenEye	Tina Turner
goodbye	Time To Say Goodbye	Brightman, Bocelli
goodgolly	Good golly miss molly	Little Richard
goodmornblues	Good morning blues	Frank Muschalle
gowest	Go West	Pet Shop Boys
griechischwein	Griechischer Wein	Udo Jürgens
grossvater	Grossvater	STS
heavensdoor	Knockin On Heaven's Door	Randy Crawford
help	Help!	Beatles
herzilein	Herzilein	Wildecker Herzbuam
holdon	Hold On	Wilson Phillips
icanfly	I Believe I Can Fly	R Kelly
icouldfly	Wish I Could Fly	Roxette
ididitagain	Oops I Did It Again	Britney Spears
ifonly	If Only	Rod Stewart
ifyoubelieve	If You Believe	Sasha
ihavenothing	I Have Nothing	Whitney Houston
imf	Mission Impossible	Adam Clayton, Larry Mullen
indy	The Raiders March	John Williams
ironic	Ironic	Alanis Morissette
itsinhiskiss	It's in his kiss	Vonda Shepard
jailhouserock	Jailhouse Rock	Elvis
joeschau	Jö schau	Georg Danzer
johnnyb	Johnny b. goode	Chuck Berry
jurassicpark	Jurassic Park	John Williams
kaktus	Mein kleiner gruener Kaktus	Comedian Harmonists
kidscene	Fremde Länder und Menschen	Schumann
kiss	Kiss	Prince
kissfromarose	Kiss From A Rose	Seal
korn-freak	Freak on a leash	Korn
lastchristmas	Last Christmas	Wham
lastdance	Save the last dance for me	The Drifters
latinolover	Latino Lover	Loona
laymedown	As I Lay Me Down	Sophie B Hawkins
leavingport	Leaving Port	James Horner
lemontree	Lemon Tree	Fools Garden
letsgetloud	Let's get loud	Jennifer Lopez
life	Life	
limb-willbeok	It'll be ok	Limp Bizkit
limp-99	9 teen 90 nine	Limp Bizkit
limp-broke	Im broke	Limp Bizkit

Identifier	Title	Interpret or Author
limp-clunk	Clunk	Limp Bizkit
limp-counterfeit	Counterfeit	Limp Bizkit
limp-faith	Faith	Limp Bizkit
limp-finger	Stink Finger	Limp Bizkit
limp-indigo	Indigo Flow	Limp Bizkit
limp-justlikethis	Just like this	Limp Bizkit
limp-lesson	Lesson learned	Limp Bizkit
limp-method	Method Man - Break stuff	Limp Bizkit
limp-n2gether	N 2 gether now	Limp Bizkit
limp-nobody	Nobody like you	Limp Bizkit
limp-nobodyloves	Nobody Loves me	Limp Bizkit
limp-nookie	Nookie	Limp Bizkit
limp-nosex	No sex	Limp Bizkit
limp-pollution	Pollution	Limp Bizkit
limp-rearranged	Re-arranged	Limp Bizkit
limp-show	Show me what you got	Limp Bizkit
limp-sour	Sour	Limp Bizkit
limp-stalemate	Stalemate	Limp Bizkit
limp-stuck	Stuck	Limp Bizkit
limp-trust	Trust	Limp Bizkit
limp-wandering	Dont go off wandering	Limp Bizkit
lorddance	Lord of the dance	Lord of the Dance
lovedwoman	Have You Ever Really Loved A Woman	Bryan Adams
lovemetender	Love Me Tender	Elvis
lovesombodysomt	Everybody loves somebody sometimes	Dean Martin
lovsisintheair	Love is in the Air	John Paul Young
macarena	Macarena	Los Del Rio
madlydeeply	Truly Madly Deeply	Savage Garden
mambofive	Mambo No 5	Lou Bega
manicmonday	Manic Monday	Bangles
matilda	Matilda	Harry Belafonte
mcgee.mps	Me and Bobby McGee	Kenny Rogers
memory	Memory	Barbara Streisand
merry	The Merry Peasant	Schumann
mindfiels	Mindfields	Prodigy
minuet	Minuet	Boccherini
missathing	I Don't Want To Miss A Thing	Aerosmith
missingyou	I'll Be Missing You	Puff Daddy
missyoucrazy	Miss You Like Crazy	Natalie Cole
mmmbop	Mmmbop	Hanson
mond	Mondscheinsonate	Beethoven
moonlightshadow	Moonlight Shadow	Mike Oldfield
morningbroken	Morning Has Broken	Cat Stevens
mountainking	In der Halle des Bergkönigs	Grieg
movingonup	Moving On Up	M-People
myheartwillgoon	My Heart Will Go On	Celine Dion
myloveyourlove	My Love is Your Love	Whitney Houston
myway	My Way	Frank Sinatra
nachtmusik	Eine kleine Nachtmusik	Mozart
nahnehnah	Nah Neh Nah	Vaya Con Dios
newyork	New York, New York	Frank Sinatra
nma-125mph	125mph	New Model Army
nma-51st	51st State	New Model Army
nma-allofthis	All of this	New Model Army
nma-betterthan	Better than them	New Model Army
nma-bigblue	Big blue	New Model Army
nma-brave	Brave new world	New Model Army
nma-dream	Western Dream	New Model Army
nma-greengrey	Green and grey	New Model Army
nma-poison	Poison Street	New Model Army
nma-questions	Stupid Questions	New Model Army
nma-race	Master Race	New Model Army
nma-war	Here comes the war	New Model Army
nocturne	Nocturne norwegisch	Secret Garden
onedaymore	One Day More	Les Miserables
onlyyou	Only You (and you alone)	The Platters
pachelbl	Canon	Pachelbel
panoptikum	Unsquare Dance	Dave Brubeck
party	Let's have a party	Wanda Jackson
peggysue	Peggy Sue	Buddy Holly
phantomopera	Phantom of the Opera	Webber
philadelphia	Streets Of Philadelphia	Bruce Springsteen
pinkpanther	Pink Panther Theme	Henry Mancini
piubellacosa	Piu Bella Cosa	Eros Ramazzotti
poweroflove	The Power of Love	Huey Lewis
pr-angels	Between angels and insects	Papa Roaches
pr-binge	Binge	Papa Roaches
pr-blood	Blood Brothers	Papa Roaches
pr-broken	Broken home	Papa Roaches
pr-deadcell	Dead cell	Papa Roaches
pr-infest	Infest	Papa Roaches
pr-lastresort	Last resort	Papa Roaches
pr-neverenough	Never enough	Papa Roaches
pr-revenge	Revenge	Papa Roaches

Identifier	Title	Interpret or Author
pr-snakes	Snakes	Papa Roaches
radetzkymarsch	Radetzkymarsch	Johann Strauß Vater
radio	Radio	The Corrs
rainbow	Over the Rainbow	Judy Garland
ratm-sun	People of the Sun	Rage against the Machine
readmymindlight	If You Could Read My Mind	Gordon Lightfoot
readmymindstars	If You Could Read My Mind	Stars on 54
rem-beyond	The great beyond	REM
rem-endoftheworld	Its the end of the world	REM
rem-orange	Orange Crush	REM
rem-religion	Losing my religion	REM
rem-stand	Stand	REM
rem-superman	Superman	REM
requiem	Requiem	Mozart
revolution	Do you hear the people sing?	Les Miserables
rhcp-angles	City of Angles	Red Hot Chili Peppers
rhcp-californication	Californication	Red Hot Chili Peppers
rhcp-dirt	I like dirt	Red Hot Chili Peppers
rhcp-easily	Easily	Red Hot Chili Peppers
rhcp-emitremmus	Emitremmus	Red Hot Chili Peppers
rhcp-getontop	Get on top	Red Hot Chili Peppers
rhcp-otherside	Otherside	Red Hot Chili Peppers
rhcp-porcelain	Porcelain	Red Hot Chili Peppers
rhcp-road	Road Trippin	Red Hot Chili Peppers
rhcp-savior	Savior	Red Hot Chili Peppers
rhcp-scartissue	Scar Tissue	Red Hot Chili Peppers
rhcp-universe	Parallel Universe	Red Hot Chili Peppers
rhcp-velvet	This velvet glove	Red Hot Chili Peppers
rhcp-world	Around the world	Red Hot Chili Peppers
risingsun	House of the Rising sun	Animals
riverdance	Riverdance	Riverdance
roadtohell	The Road To Hell	Chris Rea
rockdj	Rock DJ	Robbie Williams
rockisdead	Rock is Dead	Marilyn Manson
rocknrollkids	Rock'n Roll Kids	Paul Harrington, Charlie McGettigan
running	Keep on running	Spencer Davis Group
sanfrancisco	San Francisco	Scott McKenzie
saymyname	Say my name	Destiny's Child
schindler	Schindlers Liste	John Williams
schneib	schneibb scho obar ins Tal	Grenzlandchor Arnoldstein
schwan	Schwanensee (Scene)	Tchaikovsky
seeyouwhen	See You When You Get There	Coolio
sexbomb	Sexbomb	Tom Jones
shakespeare	The Beginning of the Partnership	Shakespeare in Love OST
she	She	Elvis Costello
shoopshoopsong	Shoop Shoop Song	Cher
sing	Sing Sing Sing	Glenn Miller
singalongsong	Sing along song	Tim Tim
sl-summertime	Summertime	Sublime
sl-whatigot	What i got	Sublime
sml-adia	Adia	Sarah McLachlan
sml-angel	Angel	Sarah McLachlan
sml-icecream	Ice Cream	Sarah McLachlan
sml-remember	I will remember you	Sarah McLachlan
sport	Es lebe der Sport	Rainhard Fendrich
starwars	Star Wars	John Williams
stormsinafrica	Storms in Africa	Enya
submarine	Yellow Submarine	Beatles
summercity	Summer in the city	Lovin' Spoonful
summerdreaming	Summer Dreaming	Kate Yanai
sunshineoflife	You Are The Sunshine Of My Life	Stevie Wonder
supertrouper	Super Trouper	A-Teens
surfusa	Surfin' USA	Beach Boys
takefive	Take Five	Dave Brubeck
talkaboutlove	Let's Talk About Love	Celine Dion
tannenbaum	Oh, Tannenbaum	
tell	William Tell Overture (conclusion)	Rossini
tellhim	Tell Him	Vonda Shepard
thecircleoflife	The Circle Of Life	Elton John
themangotree	Under The Mango Tree	Tim Tim
therose	The Rose	Bette Midler
threetimesalady	Three Times A Lady	Lionel Richie
tiffany	Breakfast At Tiffany's	Deep Blue Something
timeaftertime	Time After Time	Cyndi Lauper
timewarp	Time Warp	Rocky Horror Picture Show
togetheragain	Together Again	Janet Jackson
torn	Torn	Natalie Imbruglia
tritschtratsch	Tritsch Tratsch Polka	Strauss
unbreakmyheart	Un-Break My Heart	Toni Braxton
veronika	Veronika, der Lenz ist da	Comedian Harmonists
verve-bittersweet	Bitter sweet symphony	The Verve
verve-butterfly	Catching the butterfly	The Verve
verve-drugs	The drugs dont work	The Verve
verve-luckyman	Lucky man	The Verve

Identifier	Title	Interpret or Author
vivaforever	Viva Forever	Spice Girls
vm-4seasons	Vivaldis four Seasons	Venessa Mae
vm-bach	Bach Partita #3 in E for solo violin	Venessa Mae
vm-brahms	Brahms Scherzo in C minor	Venessa Mae
vm-classicalgas	Classical Gas	Venessa Mae
vm-red	Red violin	Venessa Mae
vm-tequlia	Tequlia Mockinbird	Venessa Mae
vm-toccata	Toccata and fugue in D minor	Venessa Mae
Walkinincohn	Walking In Memphis	Marc Cohn
walkininmemphis	Walking In Memphis	Cher
walzer	Walzer op.39 No15	Brahms
whatsawoman	What's A Woman	Vaya Con Dios
whatsup	What's Up	4 Non Blondes
whenyousay	When you say nothing at all	Ronan Keating
whitechristmas	White Christmas	Bing Crosby
wildwildwest	Wild Wild West	Will Smith
wonderfulworld	What a Wonderful World	Louis Armstrong
wonderland	Wonderland	Passion Fruit
yesterday	Yesterday	
yesterday-b	Yesterday	Beatles
yougotit	You Got It	
youlearn	You Learn	Alanis Morissette
zapfenstreich	Zapfenstreich-Hornsignale	
zarathustra	Also sprach Zarathustra	Richard Strauss
zillertaler	Zillertaler Hochzeitsmarsch	Zillertaler Schürzenjäger

Bibliography

[AHV99] E. Alhoniemi, J. Himberg, and J. Vesanto. Probabilistic Measures for Responses of Self-Organizing Map Units. In H. Bothe, E. Oja, E. Massad, and C. Haefke, editors, *Proceeding of the International ICSC Congress on Computational Intelligence Methods and Applications (CIMA '99)*, pages 286–290. ICSC Academic Press, 1999.

[Bez81] J. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms.* Plenum Press, New York, 1981.

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

[BKWW99] T. L. Blum, D. F. Keislar, J. A. Wheaton, and E. H. Wold. Method and article of manufacture for content-based analysis, storage, retrieval, and segmentation of audio information, 1999. U.S. Patent 5, 918, 223.

[Bla81] R. Bladon. Modeling the judgment of vowel quality differences. *Journal of the Acoustical Society of America*, 69:1414–1422, 1981.

[BMMS91] A. Buja, J. A. McDonald, J. Michalak, and W. Stuetzle. Interactive data visualization using focusing and linking. In *Proceedings of the IEEE Conference on Visualization*, pages 156–163, 1991.

[BNMW+99] D. Bainbridge, C. Nevill-Manning, H. Witten, L. Smith, and R. McNab. Towards a digital library of popular music. In E. Fox and N. Rowe, editors, *Proceedings of the ACM Conference on Digital Libraries (ACMDL'99)*, pages 161–169, Berkeley, CA, 1999. ACM.

[Bri74] E. O. Brigham. *The Fast Fourier Transform.* Prentice Hall, Englewood Cliffs, NJ, 1974.

[BSW96] C. M. Bishop, M. Svensén, and C. K. I. Williams. GTM: A principled alternative to the Self-Organizing Map. In C. von der Malsburg, W. von der Seelen, J. C. Vorbrüggen, and B. Sendhoff, editors, *Proceedings of ICANN'96, International Conference on Artificial Neural Networks*, volume 1112 of *Lecture Notes in Computer Science*, pages 165–170, Berlin, 1996. Springer.

[BSW97] C. M. Bishop, M. Svensén, and C. K. I. Williams. Magnification Factors for SOM and GTM Algorithms. In *Proceedings of the Workshop on Self-Organizing Maps*, pages 333–338. Helsinki University of Technology, 1997.

[Chu92] C. K. Chui. *An Introduction to Wavelets.* Academic Press, San Diego, CA, USA, 1992.

[CM98] V. Cherkassky and F. Mulier. *Learning from Data: Concepts, Theory, and Methods.* Adaptive and Learning Systems for Signal Processing, Communications, and Control. Wiley, New York, 1998.

[CPL94] P. Cosi, G. De Poli, and G. Lauzzana. Auditory Modeling and Self-Organizing Neural Networks for Timbre Classification. *Journal of New Music Research*, 23:71–98, 1994.

[DH73] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis*. Wiley, New York, 1973.

[Dix00] S. E. Dixon. A Lightweight Multi-Agent Musical Beat Tracking System. In *PRICAI 2000: Proceedings of the Pacific Rim International Conference on Artificial Intelligence*, pages 778–788, Melbourne, Australia, 2000.

[DLR77] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from inclompete data via the EM algorithm. *Journal of the Royal Statistical Society*, 39(1):1–38, 1977.

[DP96] T. Dau and D. Püschel. A quantitive model of the effective signal processing in the auditory system. *Journal of the Acoustical Society of America*, 99:3615–3622, 1996.

[DTW97] R. B. Dannenberg, B. Thom, and D. Watson. A machine learning approach to musical style recognition. In *Proceedings of the 1997 ICMC*, pages 344–347, Thessaloniki, GR, 1997.

[DWB00] M. Drobics, W. Winiwarter, and U. Bodenhofer. Interpretation of self-organizing maps with fuzzy rules. In *Proceedings of ICTAI'00*, pages 304–311, Vancouver, 2000.

[Ell96] D. P. W. Ellis. *Prediction-Driven Computational Auditory Scene Analysis*. PhD thesis, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA, 1996.

[Fas82] H. Fastl. Fluctuation strength and temporal masking patterns of amplitude-modulated broad-band noise. *Hearing Research*, 8:59–69, 1982.

[Fel55] R. Feldtkeller. Über die Zerlegung des Schallspektrums in Frequenzgruppen durch das Gehör (Division of the sound spectrum in critical bands). *Elektronische Rundschau*, 9:387–389, 1955.

[FG94] B. Feiten and S. Günzel. Automatic Indexing of a Sound Database Using Self-organizing Neural Nets. *Computer Music Journal*, 18(3):53–65, 1994.

[Foo97] J. T. Foote. Content-based retrieval of music and audio. In C. Kuo, editor, *Proceedings of SPIE Multimedia Storage and Archiving Systems II*, volume 3229, pages 138–147, 1997.

[Foo99] J. Foote. An overview of audio information retrieval. *Multimedia Systems*, 7(1):2–10, 1999.

[FR01] M. Frühwirth and A. Rauber. Self-Organizing Maps for Content-Based Music Clustering. In *Proceedings of the 12th Italien Workshop on Neural Nets (WIRN01)*, Vietri sul Mare, Italy, 2001. Springer.

[Frü01] M. Frühwirth. Inhaltsbasierte Gruppierung von Musikstücken mittels einer Self-Organizing Map (Content based clustering of music with a Self-Organizing Map). Master's thesis, Vienna University of Technology, Austria, 2001.

[Ger00] D. Gerhard. Audio Signal Classification. School of Computing Science, Simon Fraser University, 2000. Ph.D. Depth Paper.

[GLCS95] A. Ghias, J. Logan, D. Camberlin, and B. C. Smith. Query by humming Musical information retrieval in an audio database. In *Proceedings of the 3rd ACM International Conference on Multimedia*, pages 231–236, San Fancisco, CA, 1995. ACM.

[GM99] B. Gold and N. Morgan. *Speech and Audio Signal Processing: Processing and Perception of Speech and Music*. Wiley, 1999.

[Häm95] A. Hämäläinen. *Self-Organizing Map and Reduced Kernel Density Estimation*. PhD thesis, University of Jyväskylä, 1995.

[Haw90] M. Hawley. The personal orchestra. *Computing Systems*, 3(2):289–329, 1990.

[HB97] D. Hearn and M. P. Baker. *Computer Graphics, C Version*. Prentice Hall, NJ, 2nd edition, 1997.

[Hea96] C. G. Healey. *Effective Visualization of Large Multidimensional Datasets*. PhD thesis, University of British Columbia, 1996.

[Hot33] H. Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, 24:417–441 and 498–520, 1933.

[HP99] A. Härmä and K. Palomäki. HUTear - a free Matlab toolbox for modeling of human hearing. In *Proceedings of the Matlab DSP Conference*, pages 96–99, Espoo, Finland, 1999.

[IAD+92] Y. Idan, J. M. Auger, N. Darbel, M. Sales, R. Chevallier, B. Dorizzi, and G. Cazuguel. Comparative study of neural networks and non parameteric statistical methods for off-line handwritten character recognition. In *Proceedings ICANN'92*, Brighton, 1992.

[JMF99] A. K. Jain, M. N. Murty, and P. J. Flynn. Data Clustering: A Review. *ACM Computing Surveys*, 31(3):264–323, 1999.

[Jol86] I. T. Jolliffe. *Principial Component Analysis*. Springer, New York, 1986.

[Kas97] S. Kaski. *Data Exploration Using Self-Organizing Maps*. PhD thesis, Helsinki University of Technology, Department of Computer Science and Engineering, 1997.

[KHKL96] T. Kohonen, J. Hynninen, J. Kangas, and J. Laaksonen. SOMPAK: The Self-Organizing Map Program Package. Technical Report A31, Helsinki University of Technology, Department of Computer and Information Science, Espoo, Finland, 1996.

[KHLK98] S. Kaski, T. Honkela, K. Lagus, and T. Kohonen. WEBSOM-self-organizing maps of document collections. *Neurocomputing*, 21:101–117, 1998.

[Kiv96] K. Kiviluoto. Topology Preservation in Self-Organizing Maps. In *International Conference on Neural Networks (ICNN'96)*, pages 294–299, Washington, D.C., USA, 1996.

[KKL+99] T. Kohonen, S. Kaski, K. Lagus, J. Salojärvi, J. Honkela, V. Paatero, and A. Saarela. Self-Organization of a Massive Text Document Collection. In E. Oja and S. Kaski, editors, *Kohonen Maps*, pages 171–182. Elsevier, Amsterdam, 1999.

[KL96] S. Kaski and K. Lagus. Comparing Self-Organizing Maps. In C. von der Malsburg, W. von der Seelen, J. C. Vorbrüggen, and B. Sendhoff, editors, *Proceedings of ICANN'96, International Conference on Artificial Neural Networks*, volume 1112 of *Lecture Notes in Computer Science*, pages 809–814, Berlin, 1996. Springer.

[KL01] Timo Kostiainen and Jouko Lampinen. On the generative probability density model in the Self-Organizing Map. *Neurocomputing*, 2001. In press.

[KMJ95] M. A. Kraaijveld, J. Mao, and A. K. Jain. A nonlinear projection method based on Kohonen's topology preserving maps. *IEEE Transactions on Neural Networks*, 6(3):548–559, 1995.

[KNK98] S. Kaski, J. Nikkilä, and T. Kohonen. Methods for interpreting a self-organized map in data analysis. In *6th European Symposium on Artificial Neural Networks (ESANN'98)*, pages 185–190, Bruges, Belgium, 1998. D-Facto.

[KO97] K. Kiviluoto and E. Oja. S-map: A network with a simple self-organizing algorithm for generative topographic mappings. In M. I. Jordan, M. J. Kearns, and S. A. Solla, editors, *Advances in Neural Processing Systems 10*, pages 549–555. MIT Press, 1997.

[Koh82] T. Kohonen. Self-organizing formation of topologically correct feature maps. *Biological Cybernetics*, 43:59–69, 1982.

[Koh92] T. Kohonen. New developments of learning vector quantization and the self-organizing map. In *SYNAPSE'92, Symposium on Neural Networks*, Osaka, Japan, 1992. Alliances and Perspectives in Senri.

[Koh01] T. Kohonen. *Self-Organizing Maps*, volume 30 of *Springer Series in Information Sciences*. Springer, Berlin, 3rd edition, 2001.

[KOS+96] T. Kohonen, E. Oja, O. Simula, A. Visa, and J. Kangas. Engineering Applications of the Self-Organizing Map. *Proceedings of the IEEE*, 84(10):1358–1384, 1996.

[KS97] T. Kohonen and P. Somervuo. Self-Organizing Maps of Symbol Strings with Application to Speech Recognition. In *Proceedings of the Workshop on Self-Organizing Maps (WSOM'97)*, pages 2–7, Espoo, Finland, 1997.

[KVK99] S. Kaski, J. Venna, and T. Kohonen. Coloring that reveals high-dimensional structures in data. In T. Gedeon, P. Wong, S. Halgamuge, N. Kasabov, D. Nauck, and K. Fukushima, editors, *Proceedings of ICONIP'99, 6th International Conference on Neural Information Processing*, volume II, pages 729–734. IEEE Service Center, Piscataway, NJ, 1999.

[KW78] J. B. Kruskal and M. Wish. *Multidimensional Scaling.* Number 07-011 in Paper Series on Quantitative Applications in the Social Sciences. Sage Publications, Newbury Park, CA, 1978.

[LB96] A. Leonardis and H. Bischof. Robust recovery of eigenimages in the presence of outliers and occlusions. *Journal of Computing and Information Technology, CIT*, 4(1):25–36, 1996.

[Lem89] M. Leman. Symbolic and subsymbolic information processing in models of musical communication and cognition. *Interface*, 18:141–160, 1989.

[Lem94] M. Leman. Schema-based tone center recognition of musical signals. *Journal of New Music Research*, 23(2):169–204, 1994.

[Lem95] M. Leman. *Music and Schema Theory: Cognitive Foundations of Systematic Musicology.* Springer, Berlin, 1995.

[LK99] K. Lagus and S. Kaski. Keyword selection method for characterizing text document maps. In *Proceedings of ICANN99, Ninth International Conference on Artificial Neural Networks*, volume 1, pages 371–376, London, 1999. IEE.

[Log00] B. Logan. Mel Frequency Cepstral Coefficients for Music Modelling. In *International Symposium on Music Information Retrieval (MUSIC IR 2000)*, Plymouth, Massachusetts, 2000.

[LW01] M. Liu and C. Wan. A Study of Content-Based Classification and Retrieval of Audio Database. In *Proceedings of the 5th International Database Engineering and Applications Symposium (IDEAS'2001)*, Grenoble, France, 2001. IEEE.

[Mac67] J. MacQueen. Some methods for classification and analysis of multivariate observations. In L. M. Le Cam and J. Neyman, editors, *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume I of *Statistics*, pages 281–297, Berkeley and Los Angeles, CA, 1967. University of California Press.

[Mel91] D. K. Mellinger. *Event Formation and Separation in Musical Sound.* PhD thesis, Center for Computer Research in Music and Acoustics, Stanford University, Stanford, CA, 1991.

[MID96] MIDI Manufacturers Association (MMA) `www.midi.org`. *MIDI 1.0 Specification, V96.1*, 1996.

[MN95] H. Murase and S. K. Nayar. Visual learning and recogntion of 3-D objects from appearance. *International Journal of Computer Vision*, 14:5–24, 1995.

[Moo77] J. A. Moorer. On the transcription of musical sound by computer. *Computer Music Journal*, pages 32–38, November 1977.

[MR97] D. Merkl and A. Rauber. Cluster Connections – A visualization technique to reveal cluster boundaries in self-organizing maps. In M. Marinaro and R. Tagliaferri, editors, *Proceedings of the 9th Italian Workshop on Neural Nets (WIRN97)*, Perspectives in Neural Computing, pages 324–329. Springer, Vietri sul Mare, Italy, 1997.

[MSW+96a] R. McNab, L. Smith, J. Witten, C. Henderson, and S. Cunningham. Towards the digital music library: Tune retrieval from acoustic input. In *Proceedings of the 1st ACM International Conference on Digital Libraries*, pages 11–18, Bethesda, MD, USA, 1996. ACM.

[MSW96b] R. J. McNab, L. A. Smith, and I. H. Witten. Signal processing for melody transcription. In *Proceedings of the 19th Australasian Computer Science Conference*, pages 301–307, Melbourne, 1996.

[Nab01] I. Nabney. *Netlab: Algorithms for Pattern Recognition.* Advances in Pattern Recognition. Springer, Berlin, 2001.

[NL99] F. Nack and A. Lindsay. Everything you wanted to know about MPEG7 - Part 1. *IEEE MultiMedia*, pages 65–77, July/September 1999.

[Oja92] E. Oja. Self-organizing maps and computer vision. In H. Wechsler, editor, *Neural Networks for Perception*, volume I, chapter II.9, pages 368–385. Academic Press, 1992.

[PMH00] G. Peeters, S. McAdams, and P. Herrera. Instrument Sound Description in the Context of MPEG-7. In *Proceedings of the ICMC2000*, 2000.

[PTVF92] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing.* Cambridge University Press, UK, 1992.

[Rau99] A. Rauber. LabelSOM: On the Labeling of Self-Organizing Maps. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN'99)*, Washington, DC, 1999.

[RF01] A. Rauber and M. Frühwirth. Automatically analyzing and organizing music archives. In *Proceedings of the 5. European Conference on Research and Advanced Technology for Digital Libraries (ECDL 2001)*, Springer Lecture Notes in Computer Science, Darmstadt, Germany, 2001. Springer. `http://www.ifs.tuwien.ac.at/ifs/research/publications.html`.

[Rip96] B. D. Ripley. *Pattern Recognition and Neural Networks.* Cambridge University Press, Cambridge, Great Britain, 1996.

[RM99] A. Rauber and D. Merkl. The SOMLib Digital Library System. In *Proceedings of the 3rd Europ. Conf. On Research and Advanced Technology for Digital Libraries (ECDL'99)*, Paris, France, 1999. Lecture Notes in Computer Science (LNCS 1696), Springer.

[RPP00] A. Rauber, J. Paralic, and E. Pampalk. Empirical evaluation of clustering algorithms. *Journal of Information and Organizational Sciences (JIOS)*, 24(2):195–209, 2000.

[RS88] H. Ritter and K. Schulten. Kohonen self-organizing maps: Exploring their computational capabilities. In *Proceedings of the International Conference on Neural Networks (ICNN'88)*, volume I, pages 109–116, Piscataway, NJ, 1988.

[SAH79] M. R. Schröder, B. S. Atal, and J. L. Hall. Optimizing digital speech coders by exploiting masking properties of the human ear. *Journal of the Acoustical Society of America*, 66:1647–1652, 1979.

[Sam69] J. W. Sammon. A nonlinear mapping for data structure analysis. *IEEE Transactions on Computers*, 18:401–409, 1969.

[Sch00] E. D. Scheirer. *Music-Listening Systems*. PhD thesis, MIT Media Laboratory, 2000.

[SM83] G. Salton and M. J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, NY, 1983.

[SMK98] G. Smith, H. Murasse, and K. Kashino. Quick audio retrieval using active search. In *Proceedings of the 1998 ICASSP*, Seattle, 1998.

[Ter68] E. Terhardt. Über akustische Rauhigkeit und Schwankungsstärke (On the acoustic roughness and fluctuation strength). *Acustica*, 20:215–224, 1968.

[UK95] A. Ultsch and D. Korus. Automatic Aquisition of Symbolic Knowledge from Subsymbolic Neural Networks. In *Proceedings of the 3rd European Congress on Intelligent Techniques and Soft Computing, EUFIT'95*, volume I, pages 326–331, Aachen, Germany, 1995.

[Ult91] A. Ultsch. Konnektionistische Modelle und ihre Integration mit wissensbasierten Systemen (Connectionist models and their integration with knowledge based systems). Technical Report 396, Department of Computer Science, University of Dortmund, Germany, 1991.

[US90] A. Ultsch and H. P. Siemon. Kohonen's Self-Organizing Feature Maps for Exploratory Data Analysis. In *Proceedings of the International Neural Network Conference (INNC'90)*, pages 305–308, Dordrecht, Netherlands, 1990. Kluwer.

[VA99] J. Vesanto and J. Ahola. Hunting for Correlations in Data Using the Self-Organizing Map. In H. Bothe, E. Oja, E. Massad, and C. Haefke, editors, *International ICSC Congress on Computational Intelligence Methods and Applications (CIMA'99)*, pages 279–285, Rochester, New York, USA, 1999. ICSC Academic Press.

[VA00] J. Vesanto and E. Alhoniemi. Clustering of the Self-Organizing Map. *IEEE Transactions on Neural Networks*, 11(3):586–600, 2000.

[Ves00a] J. Vesanto. Neural Network Tool for Data Mining: SOM Toolbox. In *Proceedings of Symposium on Tool Environments and Development Methods for Intelligent Systems (TOOLMET 2000)*, pages 184–196, Oulu, Finland, 2000. Oulun yliopistopaino.

[Ves00b] J. Vesanto. Using SOM in Data Mining. Licentiate's thesis in the Helsinki University of Technology, 2000.

[VHAP00] J. Vesanto, J. Himberg, E. Alhoniemi, and J. Parhankangas. SOM Toolbox for Matlab 5. Report A57, Helsinki University of Technology, Neural Networks Research Center, Espoo, Finland, 2000.

[VK01] J. Venna and S. Kaski. Neighborhood preservation in nonlinear projection methods: An experimental study. In G. Dorffner, H. Bischof, and K. Hornik, editors, *Proceedings of the International Conference on Artificial Neural Networks - ICANN 2001*, pages 485–491, Berlin, 2001. Springer.

[WBKW96] E. Wold, T. Blum, D. Kreislar, and J. Wheaton. Content-based classification, search, and retrieval of audio. *IEEE Multimedia*, 3(3):27–36, 1996.

[Wei99] S. Weil. Music Analysis and Characterization. Student Project (CS 152 - Neural Networks), Harvey Mudd College, Claremont, CA `http://www.newdream.net/~sage/nn`, 1999.

[Wid01] G. Widmer. Using AI and Machine Learning to Study Expressive Music Performance: Project Survey and First Report. *AI Communications*, 14(3):149–162, 2001.

[Yan99] Wonho Yang. *Enhanced Modified Bark Spectral Distortion (EMBSD): An Objective Speech Quality Measure Based on Audible Distortion and Cognition Model*. PhD thesis, Temple University, Philadelphia, Pa, 1999.

[ZF99] E. Zwicker and H. Fastl. *Psychoacoustics, Facts and Models*, volume 22 of *Springer Series of Information Sciences*. Springer, Berlin, 2nd updated edition, 1999.

[ZFS57] E. Zwicker, G. Flottorp, and S. S. Stevens. Critical band width in loudness summation. *Journal of the Acoustical Society of America*, 29:548–557, 1957.

[ZK98] T. Zhang and C. C. J. Kuo. Hierarchical System for Content-based Audio Classification and Retrieval. In *Conference on Multimedia Storage and Archiving Systems III, SPIE*, volume 3527, pages 398–409, Boston, 1998.