Saarland University
Faculty of Natural Sciences and Technology I
Department of Computer Science

Master's Thesis

Towards Time-Adaptive Feature Design in Music Signal Processing

submitted by
Philipp von Styp-Rekowsky

submitted
March 21, 2011

Supervisor / Advisor
Priv.-Doz. Dr. Meinard Müller

Reviewers

Priv.-Doz. Dr. Meinard Müller
Prof. Dr. Michael Clausen

Eidesstattliche Erklärung

Ich erkläre hiermit an Eides Statt, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Statement under Oath

I confirm under oath that I have written this thesis on my own and that I have not used any other media or materials than the ones referred to in this thesis.

Einverständniserklärung

Ich bin damit einverstanden, dass meine (bestandene) Arbeit in beiden Versionen in die Bibliothek der Informatik aufgenommen und damit veröffentlicht wird.

Declaration of Consent

I agree to make both versions of my thesis (with a passing grade) accessible to the public by having them added to the library of the Computer Science Department.

Saarbrücken, ______________________________ ______________________________

(Datum / Date) (Unterschrift / Signature)

Acknowledgements

To begin with, thanks are appropriate to a lot of people, who helped me during my work on this thesis. Some of them are to be mentioned explicitly for their special support without which this thesis probably would not even exist.

First of all, I would like to express my deepest gratitude to my supervisor Dr. Meinard Müller. Not only did he offer me a topic so interesting that it was easy to keep up my motivation for the duration of the whole writing process, but he also was a constant source of support and advice, whenever it was needed.

Next, greatly deserved thanks go to Peter Grosche. Even while becoming a father during the period of my work, he always had an open ear to all kinds of questions I approached him with. Having him at my side helped me not to lose sight of my objective at times when I was at a loss.

Furthermore, there are my fellow students and friends, who helped me by offering various services my thesis vastly benefited from – out of pure kindness. The first person I want to point out is Frederik Dressel for supporting me in all linguistic questions. Besides him, Nanzhu Jiang and Philip Peter are to be thanked for patiently proof-reading my thesis and giving me detailed and insightful feedback.

Last but not least, I want to thank my family for their unlimited support and for being there for me whenever I needed someone to talk to. Finally, I cannot thank enough my partner Jasmin Balzert for her patience and especially for her care that accompanied me throughout my work.

Abstract

In music signal processing, it is often necessary to transform an audio signal into a representation that closely correlates to a certain musical property while being invariant to other musical aspects. The property that is captured by such a representation, which is typically referred to as *feature representation*, depends on the problem one wants to solve. For instance, in order to recognize chords that occur within a piece of music, a feature modeling the pitch of played notes is beneficial. The transformation of a music signal into a suitable feature representation, called *feature extraction*, is very common in the field of *Music Information Retrieval* (MIR).

Besides the issue of which musical characteristic to model, a second fundamental question inevitably arises when designing such a feature: How should an audio signal be segmented to capture the selected property? Current approaches use predefined windows of fixed length, that is usually empirically determined and optimized for the specific application. In this thesis, we present an adaptive method for identifying musically significant segment boundaries that are suitable for the computation of various audio features. More specifically, we incorporate rhythmic information into the feature extraction process in a modularized fashion, allowing for general applicability of the method. Extensive experiments on Western music show improvements over traditional approaches for several MIR problems.

Contents

1 Introduction

1.1 Context

In the wake of the increasing availability of music in digital formats, huge collections of musical works have come into existence. The scientific field of *Music Information Retrieval* (MIR) is devoted to facilitating access to these collections by automatically extracting information from music signals. Many applications have emerged from research in this area including automated music transcription, song identification, chord recognition, genre classification, automatic accompaniment and many more.

Music is a very complex phenomenon. The characteristics of a musical piece are not only determined by what is written in the score, namely notes and their arrangement in time, but are also greatly influenced by many other parameters such as instrumentation, dynamics, articulation and tempo. For the sake of simplicity, we will restrict ourselves in this thesis to two main dimensions of music: The vertical dimension, to which we will refer to as *spectral* dimension, comprises all aspects related to the frequencies of sounds that make up the audio signal. This includes pitches and harmonics, loudness and dynamics as well as timbral properties. The horizontal or *temporal* dimension is composed of relationships among the temporal succession of notes and silent pauses, inducing the rhythm of a musical piece. Parameters such as tempo and articulation are represented in this dimension.

To cope with the multitude of musical parameters in the context of automatic music processing, it is necessary to transform an audio signal into a representation that captures relevant key aspects while suppressing irrelevant details and variations. Deriving these representations or *features* from the music signal is crucial to making music data algorithmically accessible. It is therefore the first step in all music processing tasks. For example, the task of retrieving similar recordings from an audio database benefits from a feature representation that is invariant to details concerning the instrumentation or interpretation. Conversely, features relating to a musician's individual articulation and emotional expressiveness can be useful for the problem of artist identification.

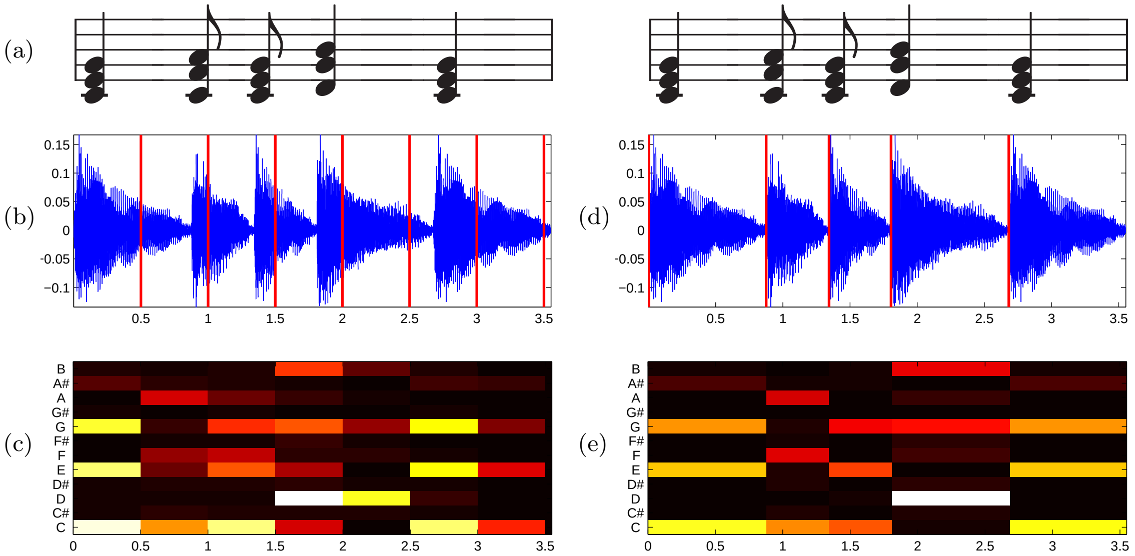


Figure 1.1: Illustration of the segmentation problem for feature extraction. **(a)** shows the score of a short cadence that was played on a piano and recorded. The recording was first segmented using fixed-length segmentation with a window length of 0.5 seconds, as shown in **(b)**, where the segment boundaries are indicated by vertical red lines. The segmented signal then served as the basis for obtaining a chroma feature representation, which is depicted in **(c)**. One can see, that the resulting feature sequence is blurred because most of the segments are influenced by more than one chord. Using onset information extracted from the audio signal, we performed adaptive segmentation on the same recording, as visible in **(d)**. The chroma representation derived from the adaptively segmented audio stream, shown in **(e)**, exhibits significantly sharper differences between the feature frames and better reflects the underlying music signal.

1.2 Problem Setting

Given the existence of a temporal dimension in music, it is clear that many musical properties vary over time. In order to capture these variations in a feature representation, it is necessary to temporally partition the audio signal into small *segments*, also referred to as *frames*. One central property we expect from these segments is that they are *homogeneous*, i.e. the measured musical aspect remains approximately the same within the segment. A feature representation can then be obtained by performing the required computations on each individual segment, yielding a sequence of feature frames.

A simple approach to segmenting the audio signal, which is also frequently used in speech processing [20], is *fixed-length segmentation*. In this method, the signal is partitioned into segments of fixed length, which is usually empirically determined and optimized for a specific application. The feature extraction process using fixed-length segmentation is sketched in Figure 1.2(a). In practice, this method is currently the one most commonly used.

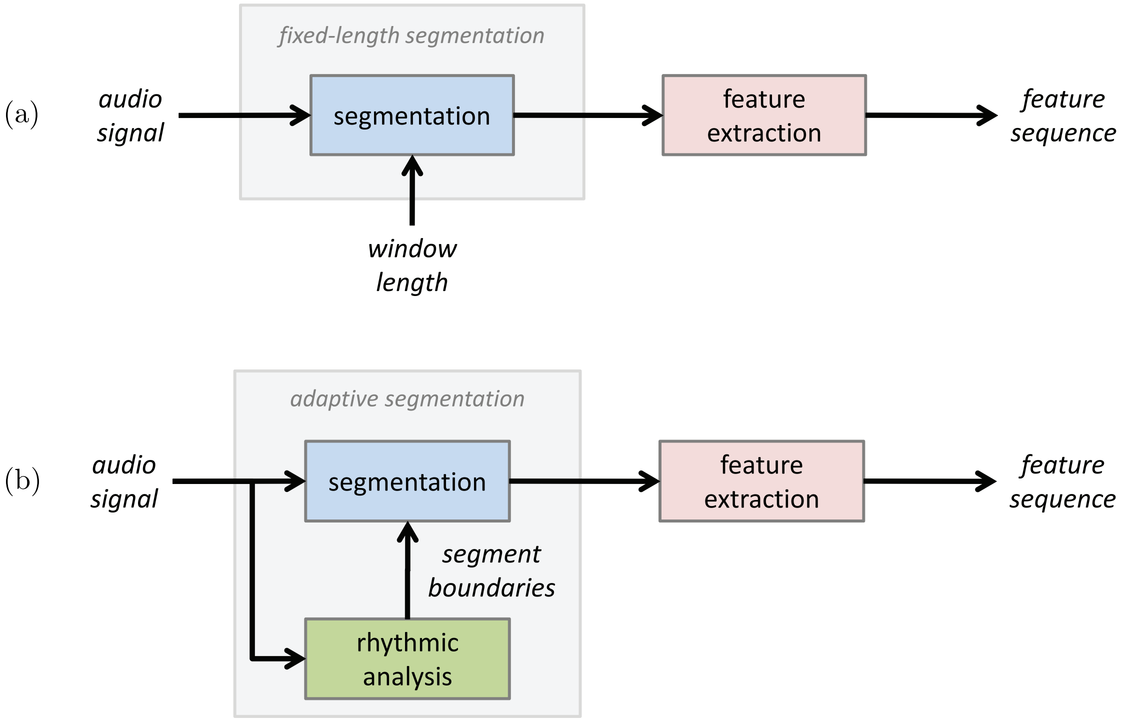


Figure 1.2: **(a)** Schematic view of the process of generating a feature sequence from an audio file using fixed length segmentation. The window length used in the segmentation step is given as a predefined parameter. **(b)** Diagram of the feature extraction process using adaptive segmentation. Here, the audio signal is segmented according to the boundaries obtained by rhythmic analysis of the audio file instead of using a predefined parameter.

However, this approach suffers from a major drawback: The boundaries of the resulting segments generally do not coincide with the changes of the captured property. This results from the fact that musical events, such as notes or percussive sounds, are usually not evenly spaced in time. Therefore, it is likely that two or more successive musical events are captured in one segment, which violates the homogeneity requirement stated above. To illustrate this problem, we applied fixed length segmentation to the waveform of the Cadence example, using a window length of 0.5s, as shown in Figure 1.1(b). A feature representation, called *chroma* representation, that models the pitches of played notes assuming octave equivalence, was then computed for each segment. As we can see in Figure 1.1(c), frames 2, 3 and 4 of the chroma representation are influenced by two chords rendering them inexpressive. This problem is typically mitigated by decreasing the window size. However, doing so introduces redundancy in the feature sequence and increases the computational cost of subsequent processing. In general, it is difficult to determine an adequate window size for fixed-length segmentation, hence this approach seems not to be well-suited for music signals.

To overcome these difficulties with fixed-length segmentation, a different technique called *adaptive segmentation*, sometimes also referred to as *rhythm based segmentation*, has been applied to music signals. This method, outlined in Figure 1.2(b), does not rely on a predefined window length, but uses rhythmical information extracted from the audio signal to identify musically relevant segment lengths and boundaries. In the following, features that are obtained using adaptive segmentation will be referred to as *time-aware* features. The signal depicted in Figure 1.1(d) has been partitioned using an adaptive segmentation technique that makes use of note onset information. The feature representation derived from this segmentation, shown in Figure 1.1(e), is homogeneous within every frame with respect to the captured chords without introducing redundancy. Thus, there is reason to believe that adaptive segmentation is an effective method for segmenting musical audio signals for the purpose of feature extraction. This thesis will therefore focus on the exploration of adaptive segmentation techniques and evaluate its effectiveness in comparison to fixed-length approaches.

1.3 Applications and Related Work

Adaptive segmentation techniques have already been applied to a wide variety of tasks in the field of music information retrieval. In particular, segmentations defined by the occurrences of a metrical pulse, most prominently the beat pulse, have proven to be especially useful for applications like *chord recognition* [3, 27, 33] and *cover song identification* [9, 11, 41]. Furthermore, adaptive segmentation has been successfully applied to *music structure analysis* [1, 5, 23, 25, 34, 35]. Also, applications like *performance analysis* [38] and *instrumentation analysis* [36] have shown to benefit from the adaptive segmentation approach.

Even though adaptive segmentation is already being employed in numerous applications, only little work has been done that examines adaptive segmentation techniques in detail. In [24], Maddage and Kankanhalli compared the effectiveness of adaptive segmentation with the traditional fixed-length approach for music content representation. To this end, the authors considered vocal and instrumental parts of musical audio signals as different music content classes. Each class was modeled by a spectral feature representation that was derived from both fixed-length segmentation with 30ms frames and adaptive segmentation based on the beat pulse. Using the inter- and intra-class distance as evaluation measure, their results indicated the superiority of the adaptive segmentation approach.

In [44], Stark et al. presented a model for beat-synchronous analysis of musical audio signals. In an informal evaluation, they compared feature sequences obtained from an adaptive beat segmentation of the signal with the classical fixed-length representations by means of a chord recognition task. Their results showed higher recognition rates for the adaptive segmentation technique.

1.4 Contribution

The main contribution of this thesis lies in the detailed analysis we conducted to evaluate the effect of using adaptive segmentation for feature extraction in comparison to traditional fixed-length approaches. While previous evaluations only considered adaptive segmentations constructed from the beat pulse, we also systematically analyze segmentations obtained from other rhythmical structures. In particular, we explore segmentation strategies that make use of local predominant pulse information. Furthermore, we examine adaptive segmentations with “gaps” that discard presumably noisy parts from the audio signal. In addition, we combine the different segmentation techniques with a wide variety of spectral features.

In view of our evaluation methodology, we introduce a novel entropy-based measure to assess the quality of the resulting feature sequences in an application-independent fashion. In order to gain further insights regarding the effect of adaptive segmentation, we also selected two prominent MIR problems, chord recognition and audio matching, to perform an application-driven evaluation.

Moreover, we carefully modeled the process of adaptive segmentation itself. As part of this formalization, we developed a flexible and extensible segmentation algorithm which can not only handle arbitrary segmentations but can also be used in conjunction with virtually any feature representation. Due to its modular design, it can easily be integrated into existing feature extraction procedures.

Last but not least, we implemented an extensible MATLAB audio player with plugin support as part of this thesis. This software does not only provide a comfortable way to listen to audio signals within MATLAB but also greatly facilitates the analysis of all kinds of feature representations. The player is documented in Appendix A.

1.5 Thesis Organization

The first two chapters of this thesis are devoted to the two main dimensions of music as described in Section 1.1. In Chapter 2, we introduce several types of spectral audio features that are frequently used in music signal processing and throughout this thesis. The temporal dimension of music will be explored in Chapter 3, focusing on the extraction of rhythmic information from the audio signal. This information is then used to construct adaptive segmentations of the audio signal, which is described in detail in Chapter 4. An extensive evaluation of the proposed technique is presented in Chapter 5, including various experiments measuring the effect of this method on two popular music processing tasks. Finally, we summarize our findings in Chapter 6.

2 Spectral Audio Features

Looking at the vertical dimension of music, every music signal can be regarded as a mixture of a set of sounds, each generated by some vibrating object such as the vocal chords of a singer or the diaphragm of a drum. All of these vibrations occur at different frequencies, inducing what humans perceive as *pitch*. The main goal of this chapter is to introduce audio features that characterize an audio signal by its spectral content and that have proven to be useful for a variety of music processing tasks. Many of the figures in this thesis rely on these feature representations for illustration purposes.

We begin by introducing some basic mathematical concepts that formalize the terms we use to describe the feature extraction process in Section 2.1. Building upon these definitions, we show how to decompose a music signal into spectral bands in Section 2.2, where each band corresponds to a pitch of the equal-tempered scale as used in Western music. This representation serves as a basis for deriving *STMSP* (*short-time mean-square power*) features that measure the local energy content of the subbands and thereby indicate the presence of certain musical notes. Assuming octave equivalence among the pitch bands, one can merge the bands that correspond to the same pitch class, obtaining a *chroma* representation of the audio signal, as described in Section 2.3. This feature is well-suited to characterize the harmonic progression of a musical recording. To increase robustness to variations of properties like dynamics, timbre and articulation, we compute short-time statistics over the energy distribution in the chroma bands, yielding *CENS* (*chroma energy normalized statistics*) features (Section 2.4). Finally, another approach to making chroma features more resilient to changes in timbre is presented in Section 2.5. Being particularly useful in audio matching and retrieval scenarios, the general idea behind *CRP* (*chroma DCT-reduced log pitch*) features is to discard timbre-related information similar to that expressed by certain mel-frequency cepstral coefficients (MFCCs).

2.1 Remarks on Feature Representations

The starting point of every feature extraction process is the *audio signal* one wishes to analyze. From a physical point of view, an audio signal describes the time-varying sound pressure of a sound wave as perceived by the human ear, progressing continuously in time. Mathematically, such an audio signal is defined as:

Definition 2.1 *An **audio signal** is a function $f : \mathbb{R} \to \mathbb{R}$, where the domain $\mathbb{R}$ represents the time axis and the range $\mathbb{R}$ the amplitude of the sound wave. Since all real-world audio signals are time-limited with a duration D, we define $T = [0, D) \subset \mathbb{R}$ as the domain of the time-limited signal and assume $f(t) = 0$ for $t \in \mathbb{R} \setminus T$. With the domain being $\mathbb{R}$, such an audio signal is also referred to as **continuous-time** (CT) signal.*

Given an audio signal, the objective of feature extraction methods is to compute feature representations or, formally, *feature sequences.*

Definition 2.2 *A **feature sequence** X is a finite ordered sequence $X = (x_1, x_2, ..., x_M)$ of feature vectors x_i from a feature space $\mathcal{X}$. In our case, $\mathcal{X}$ can be interpreted as $\mathbb{R}^d$ for some dimension d.*

Recall from Chapter 1 that a feature sequence is obtained by first segmenting the audio signal and then computing one feature vector for each individual segment. Accordingly, a single feature vector x_i carries information extracted from an interval $S_i \subset \mathbb{R}$, which we will refer to as *segment.* An ensemble of segments $\mathcal{S} = (S_1, S_2, ..., S_M)$ is called a *segmentation.* Thus, we can say that a feature sequence X is obtained on the basis of an associated segmentation S^X, that defines the mapping of feature vector indices to intervals over the temporal domain of the underlying signal.

Restricting ourselves to fixed-length segmentation for now, we define $w > 0 \in \mathbb{R}$ to be the common segment length, also called the *window length*, that is constant for all segments. In general, a fixed-length segmentation is no partition of T as we allow subsequent segments to share a common interval, the *overlap interval* $O_i = S_i \cap S_{i+1}$ of length $o \in \mathbb{R}$, which is again constant for all i. Furthermore, by definition, segments are not necessarily subsets of T. We also define the *hop size* $h = w - o$ as the difference between the start-times of adjacent segments. Using these definitions, we can construct a fixed-length segmentation of length M with window length w and hop size h by setting

$$S_n = \left[(n-1) \cdot h - \frac{w}{2} : (n-1) \cdot h + \frac{w}{2}\right) \subset \mathbb{R} \tag{2.1}$$

for $n \in [1 : M]$. To such a fixed-length segmentation, we can assign a *feature rate* $f_r = \frac{1}{h}$ which measures the number of feature vectors per second. Furthermore, we define the *overlap ratio* as the ratio $\frac{o}{w}$ of the overlap length to the window length. In Chapter 4 we will extend these definitions to incorporate adaptive segmentation.

In order to actually process an audio signal with a computer program, it is necessary to transform it into a digital representation. This transformation requires the continuous temporal domain of the audio signal to be discretized, thereby obtaining a *discrete audio signal*.

Definition 2.3 *A* ***discrete audio signal****, also called* ***discrete-time*** *(DT) signal, is function $x : \mathbb{Z} \to \mathbb{R}$ which is defined on a discrete subset of the temporal domain of a CT signal. Since the discrete signal is time-limited as well, we analogously define the $T' = [1 : N] \subset \mathbb{N}$.*

Note that, strictly speaking, the range $\mathbb{R}$ of the discrete audio signal is also discretized when represented in a digital form, however, we will ignore this detail as it plays no major role in the following.

A typical procedure to transform a CT signal into a DT signal is to sample the CT signal at equally spaced points in time, known as *equidistant sampling*. Let $f : \mathbb{R} \to \mathbb{R}$ be a CT signal and $p_s > 0 \in \mathbb{R}$ the *sampling period*, then a DT signal $x : \mathbb{N} \to \mathbb{R}$ can be obtained by

$$x(n) = f(p_s \cdot (n-1)) \ . \tag{2.2}$$

Given the sampling period p_s, or more commonly its inverse $f_s = \frac{1}{p_s}$ called the *sampling rate*, one can map time indices of a DT signal x to its continuous counterpart f by considering the injective mapping $dc : \mathbb{N} \to \mathbb{R}$ defined as

$$dc(n) = \frac{n-1}{f_s} \ . \tag{2.3}$$

Conversely, the inverse mapping $cd : \mathbb{R} \to \mathbb{N}$ from the continuous to the discrete time domain is defined as

$$cd(t) = \lfloor t \cdot f_s \rfloor + 1 \ . \tag{2.4}$$

Due to the existence of these mappings, continuous and discrete audio signals are almost equivalent. Generally speaking, the CT formulation gives the “right” interpretation of the physical phenomena, while the DT formulation is used to perform the actual computations.

DT signals and feature sequences derived from fixed-length segmentations also bear a strong resemblance. Firstly, with the DT signal having a discrete domain, it can be interpreted as a sequence of samples $x = (v_1, v_2, ..., v_N)$ with $v_i \in \mathbb{R}$, thus the definition of a feature sequence subsumes the notion of a DT signal. Furthermore, replacing the sampling rate f_s in the mapping function $dc(n)$ with the feature rate f_r of a feature sequence, the function can be used to map feature vector indices back to points in continuous time. Note that for a segmentation constructed according to Equation 2.1, the resulting point in time denotes the *center* of the segment of the corresponding feature vector. Because of this property, we call such a segmentation *zero-centered*.

Finally, note that the segmentation parameters, such as the window or hop size, are given in continuous terms i.e. in seconds, however, discrete variants of these parameters can be easily obtained using the $cd(t)$ mapping function. In the following, we will use symbols like w and h for continuous segmentation parameters, their discrete counterparts will be denoted by w' and h'. Furthermore, the symbol t will be used for the time parameter in the CT case and n in the DT case. In most plots, we will use a continuous time domain as it appears more intuitive.

2.2 Pitch Features

The ability of human hearing to distinguish between different pitches is of central importance to the perception of music. This ability is what enables us to identify a melody within a piece of music, which consists of consecutive notes at different pitch levels. Several pitches occurring simultaneously induce what we perceive as harmony, constituting one of the central elements of Western music. Hence, feature representations that capture the pitches of musical notes play a crucial role in the field of music information retrieval and are commonly used in a multitude of music processing tasks, such as chord recognition, melody tracking or audio synchronization.

In the following, we will identify a pitch according to the note numbering scheme defined in the MIDI standard, see [28] for details. This numbering scheme assigns, in increasing order, a number $p \in [1 : 127]$ to each pitch of the equal-tempered scale as used in Western music, starting at C0 and ranging up to G$^\sharp$9. For example, the middle C denoted by C4 corresponds to the number $p = 60$, whereas the concert pitch A4 has the number $p = 69$. The associated frequency $f(p)$ of a pitch p, also referred to as *center frequency*, is given by the relation

$$f(p) = 2^{\frac{p-69}{12}} \cdot 440 \tag{2.5}$$

which reveals the logarithmic nature of the pitch sensation.

The first step to obtaining a pitch feature representation is to decompose the audio signal into spectral bands where each band corresponds to a pitch. To this end, we design a suitable bandpass filter for each pitch that passes all frequencies around the respective center frequency while rejecting all other frequencies. Considering only the pitches between A0 ($p = 21$) and C8 ($p = 108$), which correspond to the keys of a standard piano, we combine 88 of these bandpass filters to an array of filters, obtaining the so-called *pitch filter bank*. The magnitude response of the resulting filter bank is plotted in Figure 2.1.

Each bandpass filter is realized using an eighth-order elliptic filter with 1 dB passband ripple and 50 dB rejection in the stopband. The bandwidth of the filter is specified by means of the so-called *quality factor* or *Q factor*, which denotes the ratio of the center frequency to the bandwidth. To cleanly separate different pitches, we chose a constant Q factor of $Q = 25$ and set the width of the transition band to be half of the width of the passband. The resulting bandwidth w_p for a filter with center frequency $f(p)$ is given by

$$w_p = \frac{f(p)}{Q} \tag{2.6}$$

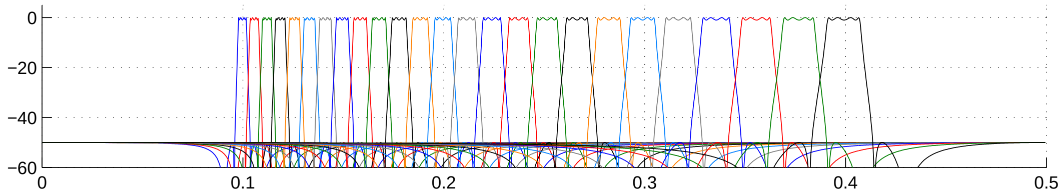


Figure 2.1: Magnitude response in dB of the pitch filter bank. Shown are the filters for the pitches $p \in [60 : 95]$ with respect to the sampling rate 4410 Hz.

thus, the bandwidth increases with for higher pitches, clearly visible in Figure 2.1. From this we obtain the cutoff frequencies of the filter as

$$\omega_{p_1} = f(p) - \frac{w_p}{2} \quad \text{and} \quad \omega_{p_2} = f(p) + \frac{w_p}{2} \tag{2.7}$$

for left and right, respectively. Note that these filter specifications are given in absolute terms and are thus only applicable to CT signals. To obtain equivalent specifications in the DT world, the described entities need to be divided by the sampling rate of the DT signal. For further details regarding the filter specifications, we refer to [28].

Furthermore, it is important to note that the proposed pitch decomposition relies on a reasonable tuning of the involved instruments according to the equal-tempered scale. Due to the passband properties of the pitch filters, deviations of up to ±25 cents from the respective center frequency of the pitch can be compensated. The logarithmic unit *cent* is used to measure musical intervals, with 100 cents corresponding to the interval between two adjacent notes. For larger deviations, a suitable tuning strategy needs to be employed that appropriately adjusts the filter bank parameters.

Finally, having obtained pitch subbands using the described pitch filter bank, we introduce our first spectral audio feature, which indicates the presence of certain musical notes in the audio signal. For this purpose, we measure the *local energy* or *short-time mean-square power* (STMSP) in each of the pitch subbands by computing the sum of the squared signal within each segment of some segmentation. More precisely, let x_p denote a DT subband signal and $\mathcal{S}$ be a segmentation with N segments S_i, then the STMSP of x at $n \in [1 : N]$ for a pitch p is defined as

$$STMSP(n, p) = \sum_{k \in S_n} |x_p(k)|^2 \ . \tag{2.8}$$

By computing the STMSP for each of the pitch subbands, we obtain a sequence of 88-dimensional feature vectors where the entries correspond to the MIDI pitches $p = 21$ to $p = 108$. For later usage, we extend each such vector by suitably adding zeros to obtain a 120-dimensional feature vector that covers the pitch range from $p = 1$ to $p = 120$. This yields the final pitch representation.

As an illustrative example, consider the Second Waltz of the Jazz Suite No. 2 by Shostakovich, which also serves as a running example in the subsequent sections. Figure 2.2(a) shows an excerpt (measures 5-12) of the piano reduced score. This excerpt corresponds to the beginning of the main theme of this piece, which occurs four times and is played in four different instrumentations. We will refer to the four occurrences as E_1 (clarinet), E_2 (strings), E_3 (trombone) and E_4. Using a recording of an orchestral version of this piece conducted by Yablonski, we visualize a pitch representation of passage E_1 in Figure 2.2(b).

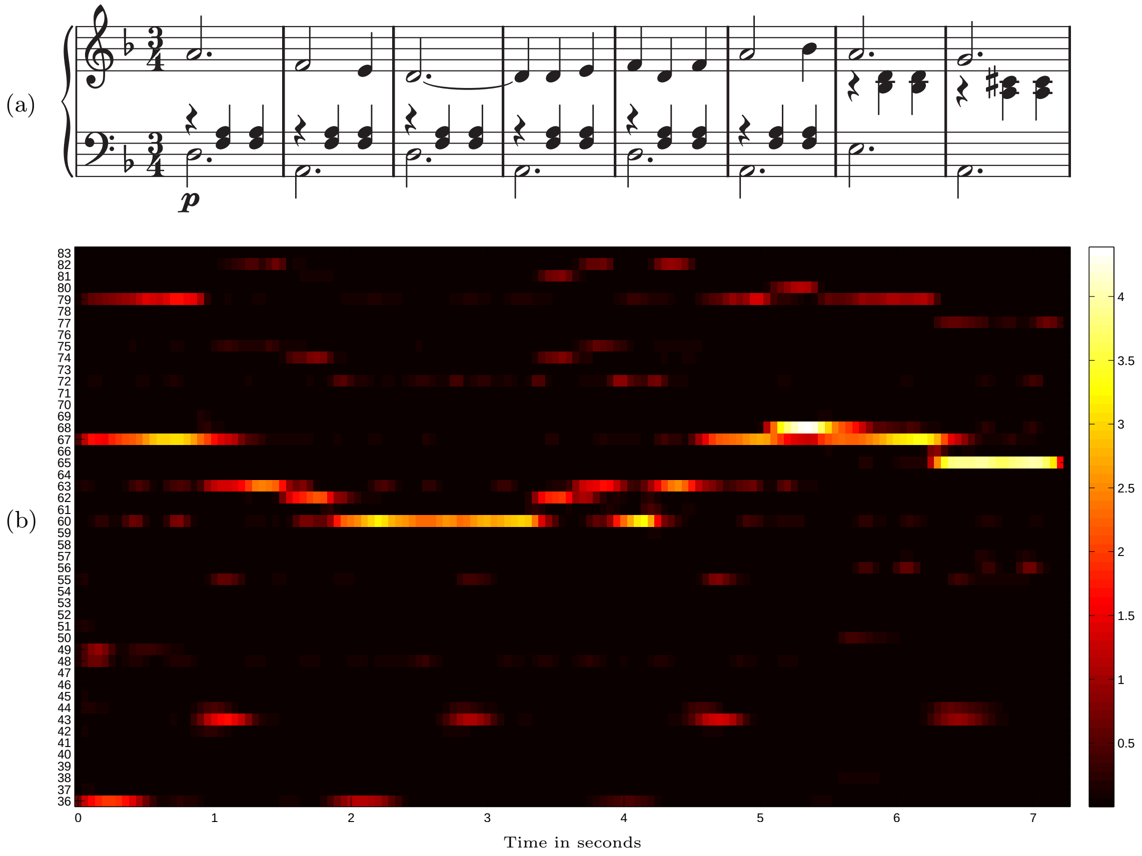


Figure 2.2: **(a)** Piano reduced score of passage E_1 (clarinet) of the Second Waltz from the Jazz Suite No. 2 by Shostakovich. **(b)** Time-pitch plot this passage computed from a recording of an orchestral version by Yablonski. The rows correspond to MIDI pitches, the time progresses along the columns. The pitch representation has been obtained using a fixed-length segmentation with window length $w = 0.2$s and an overlap ratio of $o = 0.5$, resulting in a feature rate $f_r = 10$Hz. The plot shows significant energy peaks in the pitch bands that corresponds to the melody.

All in all, the decomposition of the audio signal into pitch subbands yields musically meaningful and temporally accurate information about the spectral components of the audio signal. It is therefore very often used as a front-end signal processing step for a wide variety of MIR tasks. All of the spectral features we introduce in the following sections build upon this pitch representation.

2.3 Chroma Features

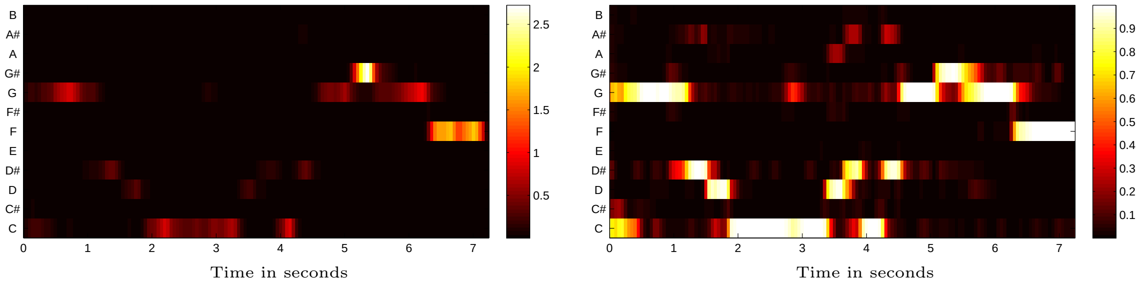
In common musical notation, the pitches of the equal-tempered scale are not identified by a MIDI note number, but in terms of the tone "color", also called *chroma*, and the *tone height*, usually written as e.g. $A4$ or $G^\sharp 6$. This notation reflects the fact that the human perception of pitch is period in the sense that two pitches are perceived as similar if they share the same chroma. This observation forms the basis of the *chroma feature* representation, which is well-suited for the analysis of music that is characterized by prominent harmonic progression. It is therefore commonly used for a wide range of MIR tasks, such as chord recognition [17, 18, 32], cover song identification [9, 19, 41] or audio matching [29, 30].

In order to obtain a chroma feature representation, we first compute a pitch representation from the audio signal as described in the previous section, and then add up all subbands that correspond to pitches with the chroma. Note that in twelve-tone equal temperament the notes $C^\sharp$ and $D^\flat$ are *enharmonically equivalent* - that is, they are identical in pitch and thus refer to the same chroma. Technically speaking, we add up pitch subbands x_i and x_j if the corresponding pitches i and j are exactly one or several octaves apart, i.e. the ratio of their center frequencies $f(i)/f(j)$ equals 2^n for some $n \in \mathbb{Z}$. In this way we reduce each 120-dimensional pitch feature vector to a 12-dimensional chroma vector, where each component represents the STMSP for the respective chroma. To increase robustness against differences in sound intensity or dynamics, we normalize each feature vector v by replacing it with $v/\left\|v\right\|_1$, where $\left\|v\right\|_1 = \sum_{i=1}^{12} |v(i)|$ denotes the ℓ^1-norm of v. The resulting representation, called *Chroma-Pitch* (CP), expresses the relative distribution of the signal's energy within the 12 chroma bands. It is visualized in form of a so-called *chromagram* in Figure 2.3(a).

A very common variant of the Chroma-Pitch feature is to apply logarithmic compression to the pitch representation prior to performing the chroma binning. This additional step is conducted to account for the fact that the human perception of sound intensity is more proportional to the logarithm of the intensity than to the intensity itself [49]. Furthermore, the compression step allows for adjusting the dynamic range of the signal to enhance the clarity of weaker transients, especially in high-frequency regions. The logarithmised version of the pitch representation is calculated by replacing each entry v_i of the feature vector $v = (v_1, v_2, ..., v_{120})$ by the value $log(C \cdot v_i + 1)$, where $C > 1 \in \mathbb{R}$ is a suitable *compression factor*. The chroma binning is then performed as previously described, yielding the so-called *Chroma-Log-Pitch* (CLP) feature representation, which is depicted in Figure 2.3(b) using a compression factor of $C = 10$.

2.4 CENS Features

Chroma-Log-Pitch are well suited to characterize the harmonic progression of a piece of music but are very sensitive to variations in local tempo, articulation and note execution,



(a)

(b)

Figure 2.3: **(a)** Chroma-Pitch feature representation of passage E_1 (clarinet) of the Shostakovich example. **(b)** Chroma-Log-Pitch feature representation of the same passage using a compression factor of $C = 10$.

as well as noise. This is especially problematic for music analysis applications like audio matching and retrieval or music synchronization. To cope with these kinds of variations, we further process the chroma features by first applying a quantization function to each chroma vector and then use a large statistics window to temporally smooth the feature sequence. The resulting feature representation is called *chroma energy normalized statistics* (CENS).

To obtain a CENS feature representation, we start with a normalized chroma feature sequence $X = (x_1, x_2, ..., x_N)$ consisting of chroma vectors $x_i \in [0,1]^{12}$. We then define a quantization function $\tau : [0,1] \to \{0,1,2,3,4\}$ as

$$\tau(a) = \begin{cases} 0 & \text{if} \quad 0 \leq a < 0.05 \\ 1 & \text{if} \quad 0.05 \leq a < 0.1 \\ 2 & \text{if} \quad 0.1 \leq a < 0.2 \\ 3 & \text{if} \quad 0.2 \leq a < 0.4 \\ 4 & \text{if} \quad 0.4 \leq a \leq 1 \end{cases} \tag{2.9}$$

and apply τ component-wise to each chroma vector. The thresholds used in this quantization step are chosen in a logarithmic fashion to account for the human perception of sound intensity. By setting chroma components that are below a 5% threshold, to zero, we introduce some robustness to noise.

To goal of the second step is to smooth out temporal micro-deviations caused by variations in note execution and articulation. The standard method to accomplish this is to convolve the feature sequence with a larger window, followed by a downsampling step to decrease the feature resolution. More precisely, the quantized chroma feature sequence is convolved component-wise with a Hann window of length $w' \in \mathbb{N}$, resulting in a sequence of 12-dimensional vectors that represent a kind of weighted statistics of the energy distribution over w' consecutive vectors. In the last step, the feature sequence is downsampled by a factor of d' and the resulting vectors are normalized with respect to the ℓ^2-norm. The final feature representation is visualized in Figure 2.4.

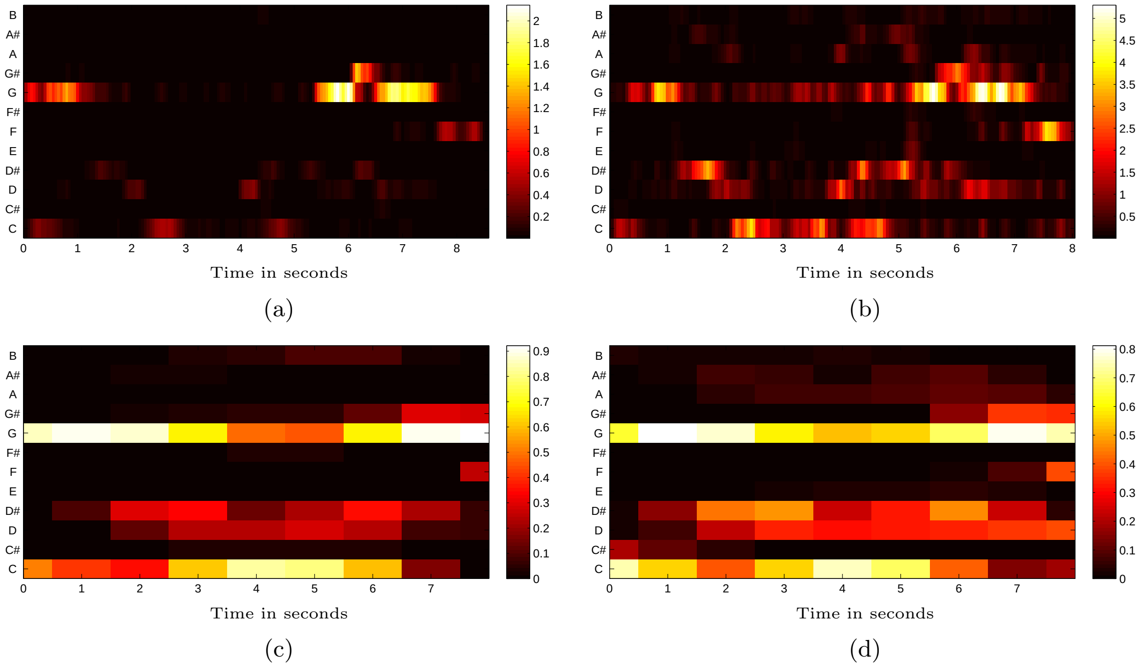


Figure 2.4: **(a)/(b)** Chroma representation of passages E_1 (clarinet) and E_2 (strings) of an orchestral version of our Shostakovich example. Due to variations in instrumentation and articulation, the two representations exhibit significant differences. **(c)/(d)** In contrast, the resulting CENS sequences computed with $w' = 41$ and $d' = 11$ are very similar.

It is important to note that the temporal smoothing step can also be expressed in terms of a segmentation. Consider a chroma feature sequence that was obtained from a fixed-length segmentation with a window size $w = 0.2$s and a hop size $h = 0.1$s, resulting in a feature rate of 10Hz. A typical realization of the CENS feature representation uses $w' = 41$ and $d' = 10$ which results in a CENS feature sequence with a feature rate of 1Hz where each vector carries information about roughly 4.1s of the underlying audio signal. Instead of employing this two-step process, it is computationally much less expensive to directly derive the chroma feature sequence from a fixed-length segmentation that uses a window size $w = 4.1$s and a hop size $h = 1$s. In conjunction with a suitable weight vector to account for the Hann window used in the convolution, one can obtain an equivalent CENS feature sequence in just one step. Furthermore, the convolution method becomes ill-defined if the underlying chroma feature sequence is obtained using an adaptive segmentation technique. Therefore, we will omit the convolution step in our experiments by setting $w' = 1$ and $d' = 1$, reducing the CENS feature computation to the quantization step.

2.5 CRP Features

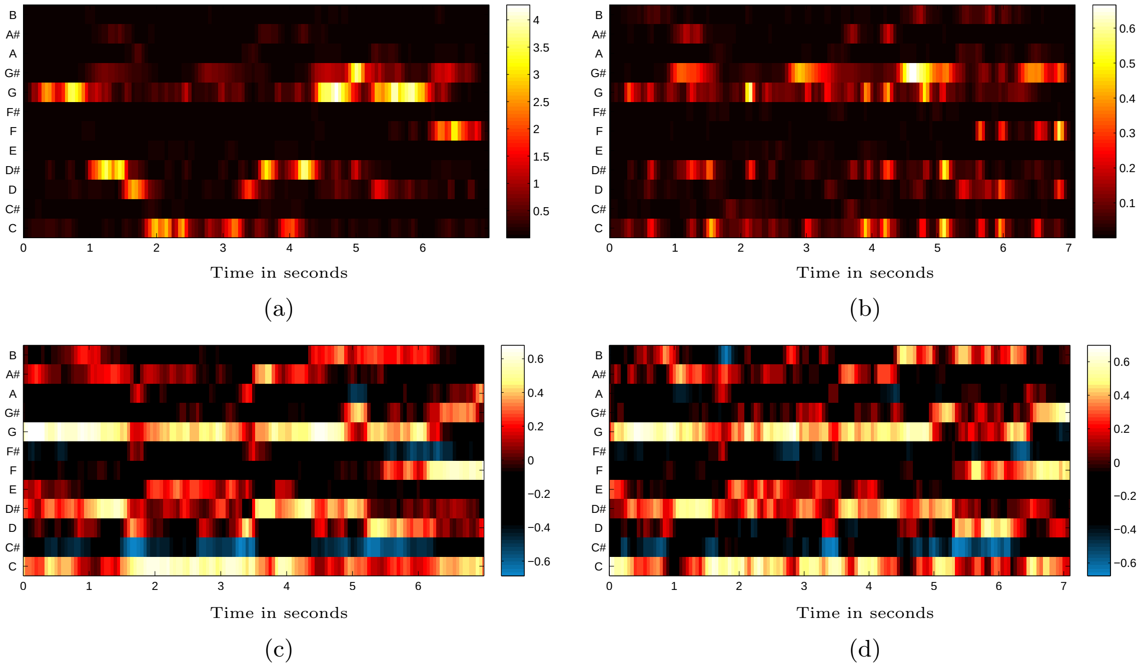
A delicate issue that is typically encountered in applications like music synchronization, audio structure analysis, cover song identification or audio matching, is to define a musically meaningful notion of similarity that can be used to compare different music excerpts. For the detection of harmony-based similarities, chroma features have proven to be useful, but suffer from the problem, that they are still sensitive to large variations in instrumentation and timbre. While CENS features primarily absorb temporal micro-deviations and increase the robustness to noise, the goal of *CRP (chroma DCT-reduced log pitch) features* is to make chroma features invariant to changes in timbre without sacrificing their discriminative power.

The derivation of CRP features is inspired by a feature representation called *mel-frequency cepstral coefficients* (MFCCs), that was originally developed in the context of speech processing [8, 37]. After finding its way into the music domain, it has been observed that MFCCs closely correlate to the aspect of timbre, making them useful for applications like musical instrument recognition [12] and genre classification [45]. Particularly the lower coefficients of a MFCC vector encode the timbre information contained in the signal. The idea behind CRP features is to discard exactly this information and thereby obtain a representation, that should exhibit a high degree of timbre invariance.

MFCC features are usually obtained in the following way. To account for the properties of the human auditory system, the signal is first decomposed into 40 nonlinearly-spaced subbands that are chosen according to the perceptually motivated mel-frequency scale. The subband signals are squared and logarithmically compressed, similar to the computation of Chroma-Log-Pitch features. The key step in the derivation is to subsequently apply a discrete cosine transform (DCT) to the subband vectors which yields the MFCCs. Finally, the upper coefficients of the DCT-transformed subband vectors are discarded, keeping only the lower coefficients, which relate to the aspect of timbre.

To derive CRP features, we incorporate the DCT transformation step into the chroma feature computation as follows. First, we compute the pitch STMSP representation as described in Section 2.2 and take the component-wise logarithm. Next, we apply the DCT transform to each of the 120-dimensional pitch vectors, resulting in 120 coefficients, which are referred to as *pitch-frequency cepstral coefficients* (PFCCs). Now, our goal of achieving timbre invariance is the exact opposite of that of MFCCs, which try to capture the timbre information. Therefore, we discard the lower $n-1$ coefficients of the PFCC vectors, given a predefined parameter $n \in [1:120]$ by setting them to zero. We then apply the inverse DCT to each of the modified PFCC vectors to obtain an enhanced pitch representation, that should not be influenced by timbral aspects of the underlying signal anymore. Finally, this pitch representation is subjected to the usual chroma binning and normalization as described in Section 2.3 The resulting features are referred to as CRP(n) (chroma DCT-reduced log pitch) features.

The increase in timbre invariance is illustrated in Figure 2.5. The chroma representations of the passages E_2 (strings) and E_3 (trombone) shown in (a) and (b) strongly deviate from each other due to the different instrumentation of the two passages. Contrary, the corresponding CRP sequences shown in (c) and (d) coincide to a much larger degree.



(a)

(b)

(c)

(d)

Figure 2.5: **(a)/(b)** Chroma representation of the passages E_2 (strings) and E_3 (trombone) in the Yablonski recording of our Shostakovich excerpt. **(c)/(d)** CRP(55) sequences of E_2/E_3.

3
Rhythmic Analysis

Rhythm is the fundamental element that gives shape to the music in the temporal dimension. Generally speaking, it denotes the arrangement of sounds and silences in time and induces a predominant pulse called *beat* or *tactus* that serves as the basis for the temporal structure of music. This chapter is devoted to techniques for the extraction and analysis of rhythmic structures found within musical audio recordings.

We begin by introducing *novelty curves* in Section 3.1, a feature representation that allows for the detection of musical accents. Essentially, this feature is obtained by computing the discrete temporal derivative of a compressed spectrum of the audio signal. Building upon the extracted note onsets, we describe a mid-level representation called *predominant local periodicity* (PLP) in Section 3.2 that reveals the local periodic nature of the musical piece. This is accomplished by determining a sinusoidal kernel for each time position that best captures the peak structure of the novelty feature, yielding an estimate for the local tatum. By suitably constraining the set of possible kernels, the PLP representation can also be used to capture the tactus or measure pulse.

3.1 Novelty Curves

The human perception of rhythm is based on inferring a regular pattern of pulses from moments of musical stress or *accents*. These accents are caused by various events in the music signal, in particular the onsets of pitched sounds, sudden changes in loudness or timbre and harmonic changes [13]. In the automatic analysis of the rhythmical structure of a musical piece, many methods imitate this process to some extend by first measuring musical accentuation and then estimating the periods and phases of the underlying pulses. In this section, we are going to present one approach to the first step of this process, that is, capturing the musical accents in a feature representation, which we call *novelty curve*.

The computation of novelty curves is based on the observation that musical accents are accompanied by sudden changes in the signal's energy and spectrum. These changes are especially distinct for instruments like the piano, guitar or percussion as they produce sounds with very sharp attacks accompanied by a broadband noise burst. The method

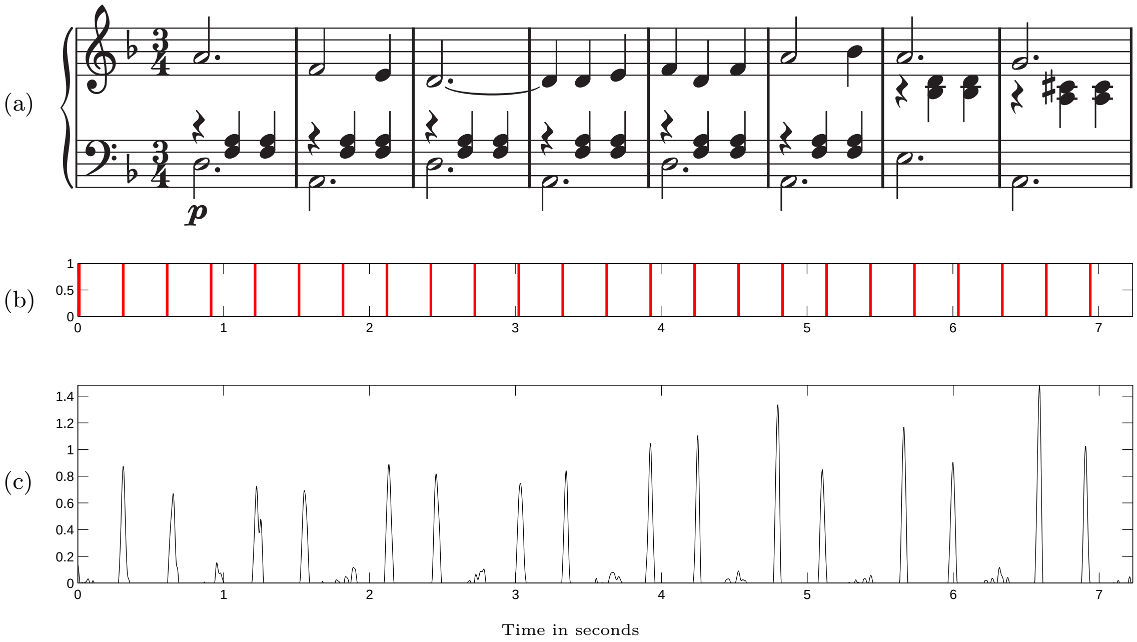


Figure 3.1: **(a)** Piano reduced score of the passage E_1 (clarinet) of the Second Waltz from the Jazz Suite No. 2 by Shostakovich. **(b)** Annotated ground truth onsets (for an orchestral recording conducted by Yablonski). **(c)** Novelty curve $\bar{\Delta}$.

we present in the following is taken from [16], other approaches exist, see for example [2, 47, 48]. First, the audio signal is decomposed into K linearly spaced frequency bands by means of a short-time Fourier transform. The result of this step is similar to the subband decomposition used to obtain the pitch representation as described in Section 2.2 but using a fixed-length segmentation to discretize the signal along the temporal domain. For the novelty curve computation however, the subband frequencies are chosen to put more emphasis on high-frequency region of the signal to capture the aforementioned noise bursts. The resulting spectral vectors are logarithmically compressed using a compression factor of $C = 1000$ to enhance weak transients in the high-frequency regions of the spectrum. To obtain the novelty curve, we basically compute the discrete derivative of the compressed spectrum. More precisely, let $Y(k,n)$ denote the kth Fourier coefficient in frame $n \in [1:N]$ of the compressed spectrum Y, then the novelty function $\Delta : [1:N-1] \to \mathbb{R}$ is defined as

$$\Delta(n) = \sum_{k=1}^{K} |Y(k,n+1) - Y(k,n)|_{\geq 0} \tag{3.1}$$

for $n \in [1:N-1]$. Here, $|x|_{\geq 0}$ denotes the *half-wave rectification* function, defined as

$$|x|_{\geq 0} = \begin{cases} x & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases} \tag{3.2}$$

for all $x \in \mathbb{R}$. By applying the rectification function, we only consider positive changes in the spectrum, discarding note offsets, which are not relevant for the musical accent feature. To eliminate spurious peaks caused by noise, we obtain the final novelty function $\bar{\Delta}$ by subtracting the local average and applying the half-wave rectification function again. An illustrative example of a novelty curve is given in Figure 3.1, showing the passage E_1 of the Shostakovich excerpt that we introduced in the previous chapter.

The peaks of a novelty curve indicate candidates for musical accents. Using an appropriate peak picking strategy, typically based on a combination of fixed and adaptive thresholding [2], we can extract the positions of significant peaks. This way, we obtain points in time where musical accents occur. However, this method is unreliable for soft and blurred onsets, as produced by bowed string instruments or the human voice. In this case, novelty curves tend to be noisy which makes it hard to distinguish musically meaningful peaks from spurious ones. For this reason, we further process the novelty curve to take the underlying periodicities into account, which we describe in the next section.

3.2 Predominant Local Periodicity

Rhythm is a hierarchical structure, that is defined by means of periodic pulses at different levels (time scales). One typically considers three metrical levels. As previously mentioned, the most prominent level is the beat, also called *tactus*. It serves as the basis for defining the *tempo* of a musical piece, which is typically measured in *beats per minute* (BPM). The *measure* pulse, which corresponds to the coarsest level of the rhythmical hierarchy, is induced by regularly recurring patterns of stressed and unstressed beats. In Western music, this pattern is reflected by grouping subsequent beats into bars or *measures*. In common music notation, the number of beats per measure (typically 3 or 4) is indicated by a time signature and often remains constant throughout the piece. The measure pulse is then defined by considering only the first beat of each measure. The finest metrical level is constituted by the *tatum* pulse. The term tatum stems from "temporal atom" and refers to the fastest repetition rate of musically meaningful accents occurring in the signal. Thus, all note onsets approximately occur at tatum pulse positions. Furthermore, in the majority of cases, the period of the beat and measure pulse are integer multiples of the tatum pulse period.

While novelty curves are useful for the detection of musical accents, they do not well reflect the aforementioned periodic pulses that a human listener would infer from these accents. Because of their central importance to the perception of rhythm, we need a way to extract these pulses from the audio signal. To this end, we present a method that aims at detecting the *predominant local periodicity* (PLP) of accents in the music signal, which typically corresponds to the tatum pulse.

Many approaches exist for analyzing musical accent signals with respect to periodic components. Prominent examples are autocorrelation methods [10], comb filter techniques [39] and inter-onset interval analysis [40]. However, these approaches encounter difficul-

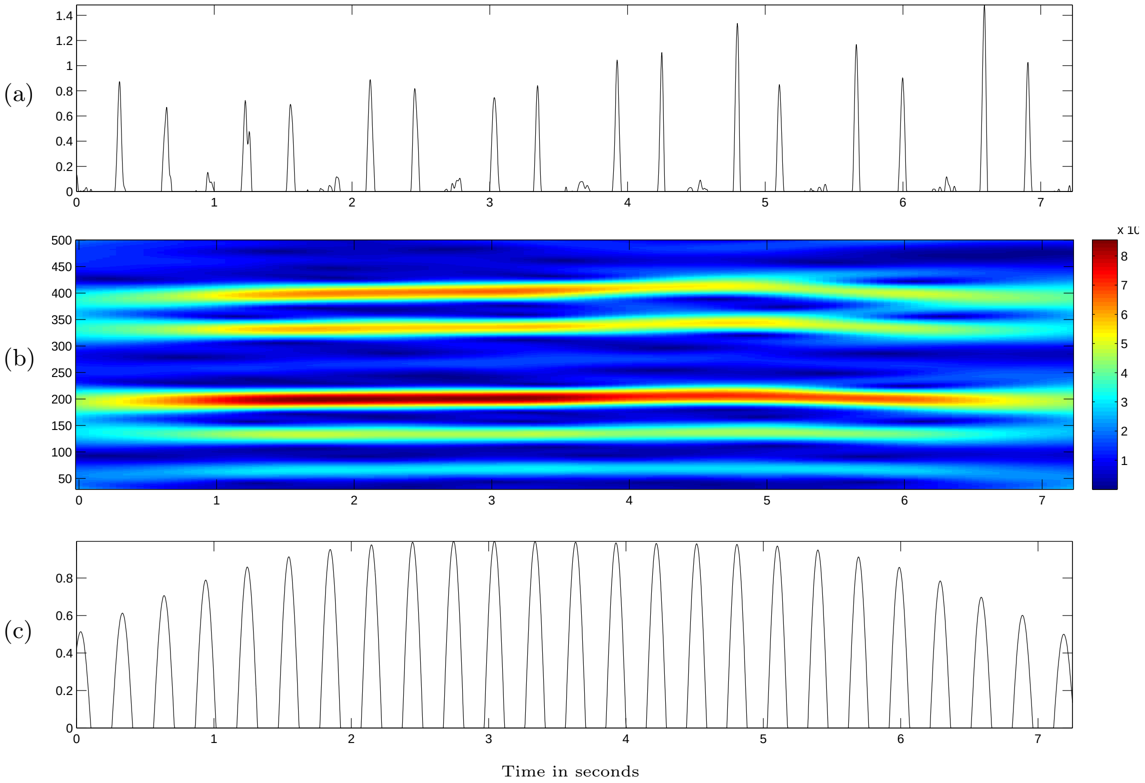


Figure 3.2: **(a)** Novelty curve $\bar{\Delta}$ for the passage E_1 of the Shostakovich Waltz. **(b)** Magnitude tempogram of E_1 computed with $w = 4s$ and $h = 0.05s$. **(c)** Resulting PLP curve.

ties when dealing with accent signals that are noisy or exhibit spikes that are irregularly spaced over time, which holds for the previously described novelty curves. To cope with these irregularities, the PLP approach, which was first presented in [16], uses smoothly spread sinusoids to detect the locally distorted quasiperiodic patterns contained within the novelty curve. By accumulating the local periodicity information to form a single function, we obtain a robust mid-level representation called *PLP curve* that reveals the local periodic nature of the underlying music signal.

Given a possibly noisy novelty function, the PLP curve is computed as follows. First, we investigate the local periodicity of the novelty curve $\bar{\Delta}$ of length N using a short-time Fourier transform with a suitable set of frequency parameters. In our experiments, we mostly use the set that corresponds to the musical tempi between 30 and 500 BPM. These bounds are motivated by the observation, that only events with a temporal separation between 120 milliseconds and 2 seconds contribute to the perception of rhythm [4]. For the temporal segmentation of the novelty curve, we use a fixed-length analysis window with a large overlap ratio whose size is chosen to capture local tempo changes. In our experiments, we use a Hann window of length $w = 4s$ and a hop size $h = 0.05s$. This way, we obtain a time-pulse representation called *tempogram* that indicates the strength of a local pulse over time. Note that a tempogram is complex-valued and contains information about both the magnitude as well as the phase of a pulse. Figure 3.2(b) shows the magnitude part of such a tempogram, which was obtained by analyzing the novelty curve computed from the Shostakovich example in the previous section. All note onsets in this example are evenly spaced, inducing a constant pulse corresponding to roughly 200 BPM. This pulse is clearly visible in the tempogram, which exhibits a significant peak at 200 BPM throughout the piece.

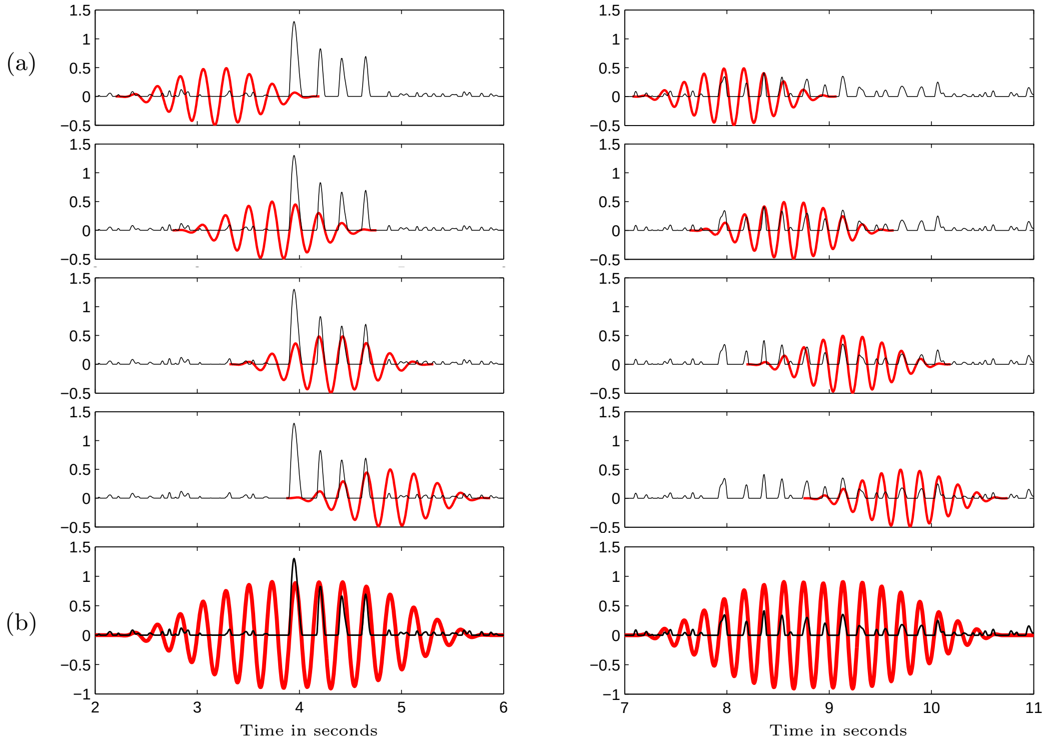


Figure 3.3: **(a)** Optimal sinusoidal kernels (red) for various time positions for two parts of a novelty curve (black) obtained from the first 12 measures of Beethoven's Symphony No. 5. **(b)** Accumulation of the kernels shown in (a). Applying half-wave rectification yields the PLP curve. Figure reproduced from [16].

While the tempogram reveals the strength and phase of a local pulse, we are more interested in a representation that allows for the determination of the exact time positions at which the individual pulses occur. To this end, we first identify for each time position the pulse frequency that maximizes the magnitude of the tempogram. Taking the phase information given by the tempogram into account, we can construct a sinusiodal kernel for each time position that best explains the local periodic nature of the novelty curve, as visualized in Figure 3.3. To increase the robustness of the kernel estimation for novelty curves with strongly corrupted pulse structures, we finally accumulate the kernels over all time positions to form a single function. This function, that denotes the final PLP curve, is obtained by summing up all kernels for each time position followed by half-wave rectification. The resulting curve is depicted in Figure 3.2(c). As it turns out, the PLP curve is robust to outliers and yields musically meaningful pulse information even when given poor onset information. For further details, we refer to [16].

The peaks of the PLP curve indicate the time positions of the individual musical events that make up the local predominant pulse. In most cases, the predominant pulse corresponds to the tatum, however, the semantic level may change over time. Thus the predominant pulse can also refer to the tactus or measure pulse. We illustrate this behavior with the help of a different excerpt (measures 25-35) from the Shostakovich Waltz, see Figure 3.4(a) for a piano reduced score. Here, in the first four measures, the predominant pulse corresponds to the tactus, as musical accents occur only at quarter note positions. The eighth notes in subsequent measures cause the predominant pulse to change to the tatum level, which is captured by the PLP curve shown in Figure 3.4(e). However, in some cases, it is desirable to prohibit the PLP curve from switching between different semantic levels. To this end, we constrain the set of tempo parameters used in the computation of the tempogram, incorporating prior knowledge about the tempo of the piece. For example, to extract only the tactus pulse (roughly 200 BPM) from the Shostakovich excerpt shown in Figure 3.4(a), we use a constrained set of tempo parameters that covers the tempo range between 141 BPM and 283 BPM. This range corresponds to a tempo octave around the annotated ground truth tempo of 200 BPM. The resulting tempogram and PLP curve are shown in Figure 3.4(f) and (g) respectively.

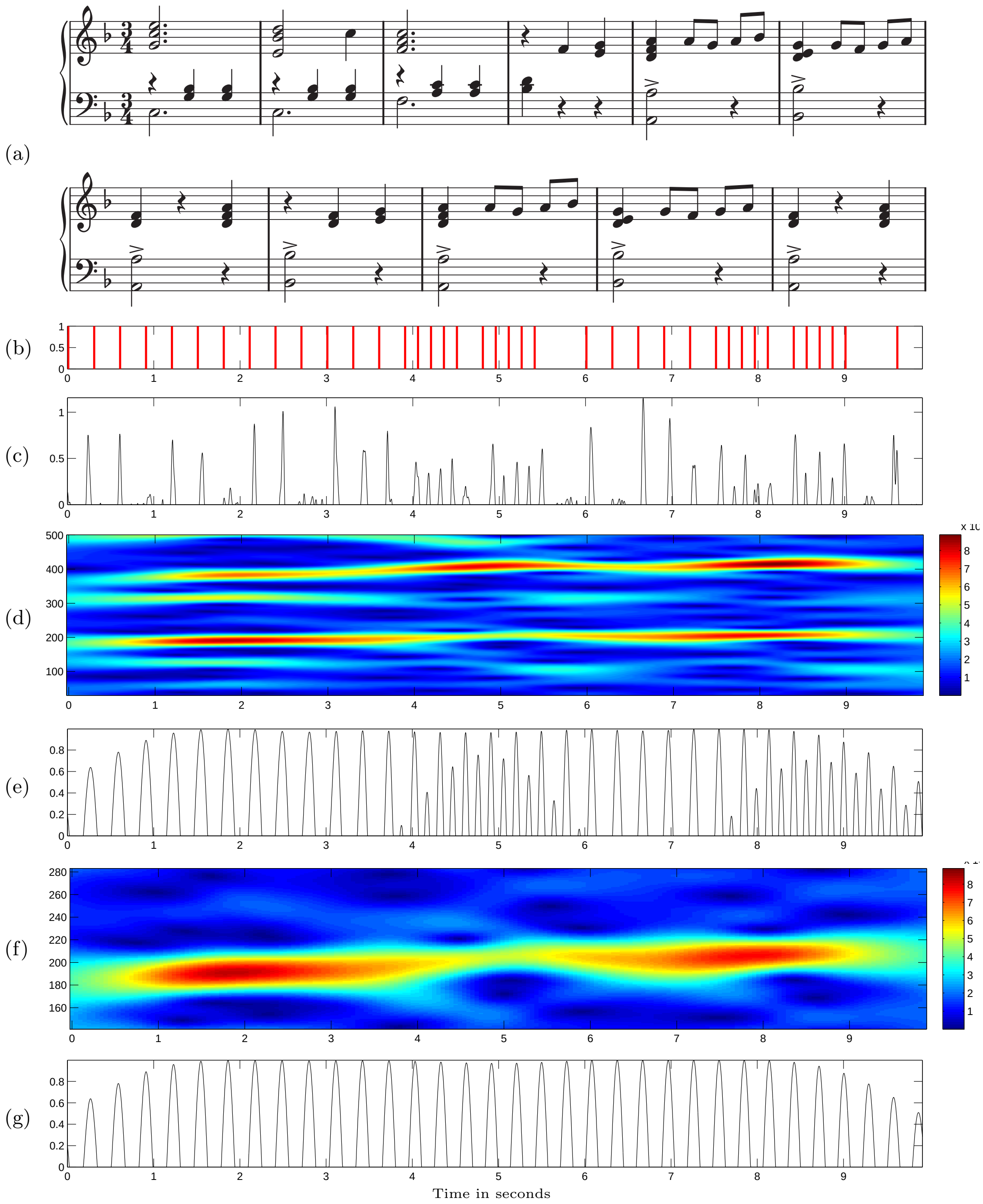


Figure 3.4: **(a)** Piano reduced score of the measure 25-35 of the Shostakovich Waltz. **(b)** Annotated ground truth onsets. **(c)** Novelty curve. **(d)** Magnitude tempogram with tempo range 30 to 500 BPM. **(e)** PLP curve derived from tempogram (d). **(f)** Constrained magnitude tempogram with tempo range 142 to 283 BPM. **(g)** PLP curve derived from tempogram (f).

4
Adaptive Segmentation

Having analyzed the rhythmical structure of the musical piece, we can now use this information for adaptively segmenting the audio stream. Our goal is to perform segmentation in a way that any subsequent feature extraction steps yield sequences of frames that correspond to musical events instead of generating a feature representation consisting of frames with no musically meaningful interpretation. In this chapter, we will focus on the actual segmentation algorithm and give details of its implementation.

In Section 4.1 we show how to construct adaptive segmentations of the audio signal using the information gathered in the rhythmic analysis step. We present a practical approach to adaptive segmentation in Section 4.2, where we introduce a novel segmentation algorithm that can not only be used in conjunction with virtually any spectral feature extractors but can also handle audio streams that have already been arbitrarily segmented. We aim at further improving the final feature representations by describing an extensions to our algorithm in Section 4.3 that permits the removal of presumably noisy parts from the audio stream.

4.1 Adaptive Segmentation

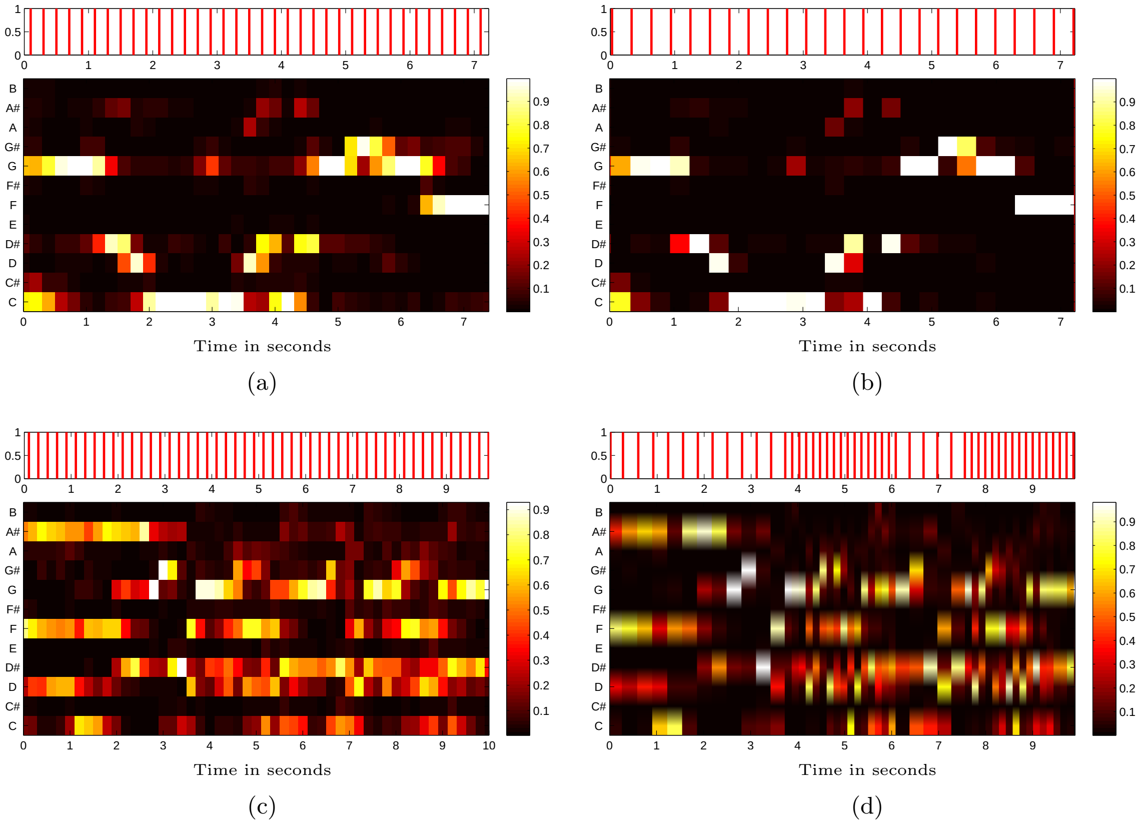
In Chapter 2, we have seen how to obtain feature sequences that characterize an audio signal by its spectral content. Methods that extract information about the temporal and rhythmical structure of the signal, have been presented in Chapter 3. So far, these techniques have been applied individually. We will now show how to combine these approaches in the feature extraction process to obtain an improved feature representation. More precisely, we will use the rhythmical information to construct an adaptive segmentation of the audio stream that is then used to obtain the spectral feature representation.

In general, the rhythmical information we have gained about the signal, is given by a series $R = (r_1, r_2, ...r_L)$ of length L, that consists of the time positions r_i of relevant rhythmical events. In our experiments, such a series of time positions is obtained by performing peak picking on various PLP curves as well as using human reference annotations for beat positions.

Recall from Section 2.1 that a segment S was defined as an interval $S = [s, t) \subset \mathbb{R}$ and a segmentation $\mathcal{S} = (S_1, S_2, ..., S_N)$ denotes as series of segments S_i. Given a series of time positions R, we can construct an adaptive segmentation $\mathcal{S}$ by simply using the time positions r_i as segment boundaries, thus

$$S_i = [r_i, r_{i+1}) \subset \mathbb{R}. \tag{4.1}$$

for $i \in [1 : L - 1] \subset \mathbb{N}$. That way, we obtain a non-overlapping segmentation of the audio stream, where each segment boundary coincides with some rhythmical event.



(a) (b)

(c) (d)

Figure 4.1: **(a)/(b)** Chromagrams of the passage E_1 of the Shostakovich Waltz. The boundaries of the underlying segmentations are indicated above. The chromagram (a) has been obtained via fixed-length segmentation with window length $w = 0.2$s whereas (b) was obtaind using an adaptive segmentation constructed from PLP curve peaks. **(c)/(d)** Chromagrams of the measures 25-35 of the Shostakovich Waltz. The chromagram (c) was obtained using the same fixed-length segmentation parameters as in (a). The adaptive segmentation underlying the chromagram in (d) reflects the changes in the local predominant pulse of the musical piece.

We illustrate the effect of adaptive segmentation in Figure 4.1 by means of the two excerpts from the Shostakovich Waltz we introduced in previous chapters. The normalized chroma feature representation depicted in (a) corresponds to the passage E_1 (see Figure 3.1(a) for the score) and was derived using a fixed-length segmentation with a window length of 0.2 seconds and no overlap. The segment boundaries are indicated above the chromagram. While the melody of the excerpt is relatively well visible in this chromagram, it is difficult to see where the individual notes start and end because the note onsets and offsets are blurred. In contrast, the chromagram depicted in (b), which corresponds to the same passage, was obtained from an adaptive segmentation using the peaks of the PLP curve shown in Figure 3.2(c) as segment boundaries. Even though the segments have approximately the same length, they are accurately aligned with the note onsets in the signal. This results in sharp and well developed peaks in the chromagram that facilitate the identification of the individual notes. Thus, this representation better reflects the musical content of the underlying signal even though it consists of less frames than the fixed-length representation.

A different example is shown in Figure 4.1(c) and (d), see Figure 3.4(a) for the corresponding score. Here, the chromagram depicted in (c) has again been derived from a fixed-length segmentation using the same settings as in (a). The individual notes, especially the eighth notes in the second part of the excerpt are indistinguishable from each other. This is not the case for the chromagram visualized in (d). This chroma representation has been computed using an adaptive segmentation constructed from the PLP curve shown in Figure 3.4(e). The segment lengths are appropriately adjusted according to the predominant local pulse given by the PLP curve. Hence, the resulting chromagram clearly reflects the rhythmical structure of the excerpt and reveals the eighth notes occurring in the signal.

It should be noted that adaptive segmentations can – in principle – be constructed arbitrarily. While a construction according to Equation 4.1 is suitable in most MIR scenarios, it is also possible to create segmentations with, for example, overlapping segments. One could also construct a segmentation with “gaps” to exclude unwanted or irrelevant parts of the signal, which we elaborate in Section 4.3. In the end, the choice of a suitable adaptive segmentation depends entirely on the respective application.

4.2 Segmentation Algorithm

The construction of the adaptive segmentation as given above is the central step to obtain time-aware feature sequences. Given such a segmentation, the feature extraction steps as described in Chapter 2 can be straightforwardly applied to yield time-aware features. However, from a practical point of view, many applications often require a flexible and computationally inexpensive procedure to adjust the time resolution of a feature sequence. This is particularly relevant when conducting experiments for MIR research. The CENS feature representation we introduced in Section 2.4 already provides a simple mechanism to dynamically adjust the feature rate: Instead of re-computing the pitch representation, which is by far the computationally most expensive step in the CENS derivation, the pitch representation is first computed with a high temporal resolution and then smoothed and downsampled to yield the desired feature rate. The pitch representation is saved and can be reused if a different feature rate is required. However, as previously mentioned, this mechanism only works for features derived from a fixed-length segmentation and is thus not applicable to time-aware features sequences.

To this end, we now introduce a novel method to transform an arbitrary feature sequence X that has been created using a segmentation S^X into a new feature sequence Y with respect to a different segmentation S^Y. In the typical use case, S^X denotes a fixed-length segmentation with small window length and S^Y is an adaptive segmentation created from rhythmical information. In general, the method works with arbitrary adaptive segmentations with the restriction that the input segmentation S^X must be non-overlapping.

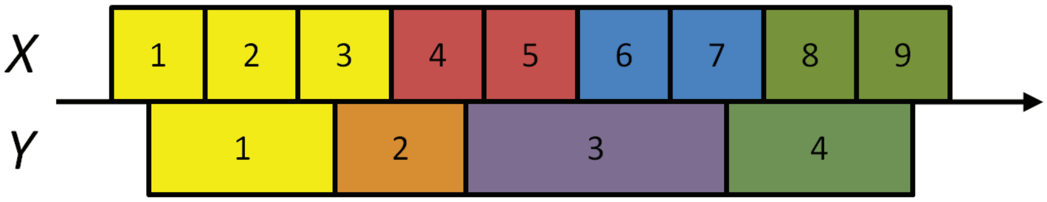


Figure 4.2: Illustration of our segmentation algorithm. The fixed-length input feature sequence X is transformed into the time-aware output feature sequence Y.

Intuitively, our method works as follows. We construct the output feature sequence $Y = (y_1, y_2, ..., y_M)$ in a two-step process: Firstly, we identify for each output feature vector y_i a subsequence of input feature vectors in the input feature sequence $X = (x_1, x_2, ..., x_N)$, that temporally correspond to y_i. Secondly, this sequence of input feature vectors is aggregated to form one single output feature vector y_i. Figure 4.2 illustrates the method for the typical use case. Here, the input feature sequence X was derived using a fixed-length segmentation and is to be transformed into a time-aware output feature sequence Y on the basis of an adaptive segmentation. For illustration purposes, the individual feature vectors have been color-coded where different colors indicate different values. The output feature vector y_1 is influenced by the input feature vectors x_1, x_2 and x_3 which all have the same "value" yellow, thus y_1 attains a yellow value as well. The second output frame y_2 is roughly equally influenced by x_3 and x_4, which are yellow and red respectively. Hence, the color orange, which is a composite color consisting in equal parts of yellow and red, is assigned to y_2. The general idea behind the computation of the output feature vectors y_i is that the contribution of each relevant input feature vector x_j to the value of y_i is proportional to the *overlap ratio* between the two corresponding segments S_i^Y and S_j^X. We have already defined the overlap ratio for fixed-length segmentations in Section 2.1, for adaptive segmentation however, a more general definition is required. Using our set notation for segments, the overlap ratio $o(S_i, S_j)$ of two segments S_i and S_j can simply be defined as

$$o(S_i, S_j) = \frac{|S_i \cap S_j|}{|S_i|} \ . \tag{4.2}$$

An alternative and more practical definition of the overlap ratio can be formulated, when a segment S is regarded as a tuple $S = (s, t)$ with $s, t \in \mathbb{R}$ being the start and end time of the segment. The overlap ratio can then be computed as

$$o(S_i, S_j) = \begin{cases} 0 & \text{if } s_i \geq t_j \vee s_j \geq t_i \\ \frac{\min(t_i,t_j)-\max(s_i,s_j)}{t_i-s_i} & \text{otherwise} \end{cases} . \tag{4.3}$$

To construct the output feature sequence Y, we can now use the overlap ratio to identify for each y_i those frames in X that contribute to the value of y_i. More precisely, let y_n denote the output feature vector we want to compute. To this end, we determine the sequence of feature indices $R^n = (r_1^n, r_2^n, ..., r_L^n)$ with $r_i^n \in [1 : N]$ that satisfies the following two conditions.

(i) Monotonicity condition: $r_i^n < r_{i+1}^n$ for $i \in [1 : L - 1]$

(ii) Relevance condition: $o(S_n^Y, S_{r_i}^X) > 0$ for $i \in [1 : L]$

The monotonicity condition (i) ensures, that the order of the feature vectors in the resulting subsequence $X(R^n) = (x_{r_1^n}, x_{r_2^n}, ..., x_{r_L^n})$ is preserved. As indicated by the relevance condition (ii), we only consider a feature vector x_i to be relevant for computation of y_n, if the corresponding segments S_i^X and S_n^Y have non-zero overlap.

To obtain the final value of y_n, we aggregate the subsequence $X(R^n)$ to form a single feature vector. Generally speaking, the choice of a suitable aggregation method depends on the nature of the feature space $\mathcal{X}$. For the spectral features we introduced in Chapter 2, we have $X = \mathbb{R}^d$ for some dimension d. To honor the influence of each input feature vectors on y_n, we first define a weight vector w^n of length L as

$$w_l^n = o(S_n^Y, S_l^X) \tag{4.4}$$

for $l \in [1 : L]$. The output feature vector y_n is then computed as the weighted sum over all $X(R^n)$, thus

$$y_n = \sum_{l=1}^{L} w_l^n x_{r_l^n} . \tag{4.5}$$

That way, each input feature vector $x_{r_i^n}$ contributes only a fraction equal to the overlap ratio of the corresponding segments to the value of y_n. By computing y_n for all $n \in [1 : M]$, we obtain the final output feature sequence Y.

The proposed method constitutes a flexible and computationally inexpensive way to compute time-aware feature sequences. Due to its ability to transform feature sequences derived on the basis of fixed-length segmentation into time-aware representations, it can be easily integrated into existing feature extractors. By choosing a suitable aggregation method, it can be applied to feature sequences from arbitrary feature spaces. For example, feature vectors from a discrete feature space could be aggregated with the help a majority vote aggregation function. Furthermore, by appropriately modifying the weight vector w_l, it is possible to incorporate a window function, such as a Hann window, into the aggregation process. Our MATLAB implementation of this algorithm is documented in Appendix A.

4.3 Noise Removal

As described in Section 3.1, the detection of musical accents is based on the observation, that sounds with sharp attacks cause a broadband noise burst, especially for instruments like the piano, guitar or percussion. This phenomenon is illustrated in Figure 4.3(a), which shows a pitch decomposition of a chromatic scale played on a piano. The noise bursts are clearly visible as vertical lines at the beginning of each note. On a side note, this figure also reveals the fact, that striking a single key on the piano produces a complex sound comprised of a mixture of different frequencies. Especially for lower pitches, most of the energy is contained in the frequency bands corresponding to the higher harmonics of the pitch.

While these noise bursts are useful for musical accent detection, they cause undesired artifacts in spectral feature representations as they introduce spurious energy in a wide range of frequency bands. However, we can identify the time positions of the noise bursts by using the rhythmical analysis techniques we have introduced in Chapter 3. These time positions can then be used as segment boundaries to construct an adaptive segmentation of the signal that has “gaps” in the areas around the noise bursts, thereby excluding them from further consideration. To facilitate the construction of such a segmentation, we slightly modify our segmentation algorithm by introducing a parameter $\alpha \in [0,1] \subset \mathbb{R}$. This parameter has the effect, that we consider only the fraction α of the length of each output segment S_i^Y for the selection of relevant input feature vectors. More precisely, we preprocess every output segment $S_i^Y = (s_i, t_i)$ of length $l_i = t_i - s_i$ to yield a modified segment $\bar{S}_i^Y$ of length $\bar{l_i} = \alpha * l_i$ by setting

$$\bar{S}_i^Y = (s_i + \frac{1-\alpha}{2} * l_i, t_i - \frac{1-\alpha}{2} * l_i) \; . \tag{4.6}$$

With this construction, the time positions corresponding to the center of the segments are preserved. The segment boundaries are shifted by $\frac{1-\alpha}{2} * l_i$ towards the center of the segment to exclude the parts of the signal that are close to the noise bursts. The effect of this method is illustrated in Figure 4.3(c). Here, the pitch representation of the signal has been obtained using an adaptive segmentation constructed from PLP curve peaks and $\alpha = 0.5$. In contrast to the fixed-length feature shown in Figure 4.3(a), no noise bursts or artifacts are visible. For comparison, the pitch representation depicted in Figure 4.3(b) was computed with $\alpha = 1.0$ and exhibits a large number of spurious artifacts.

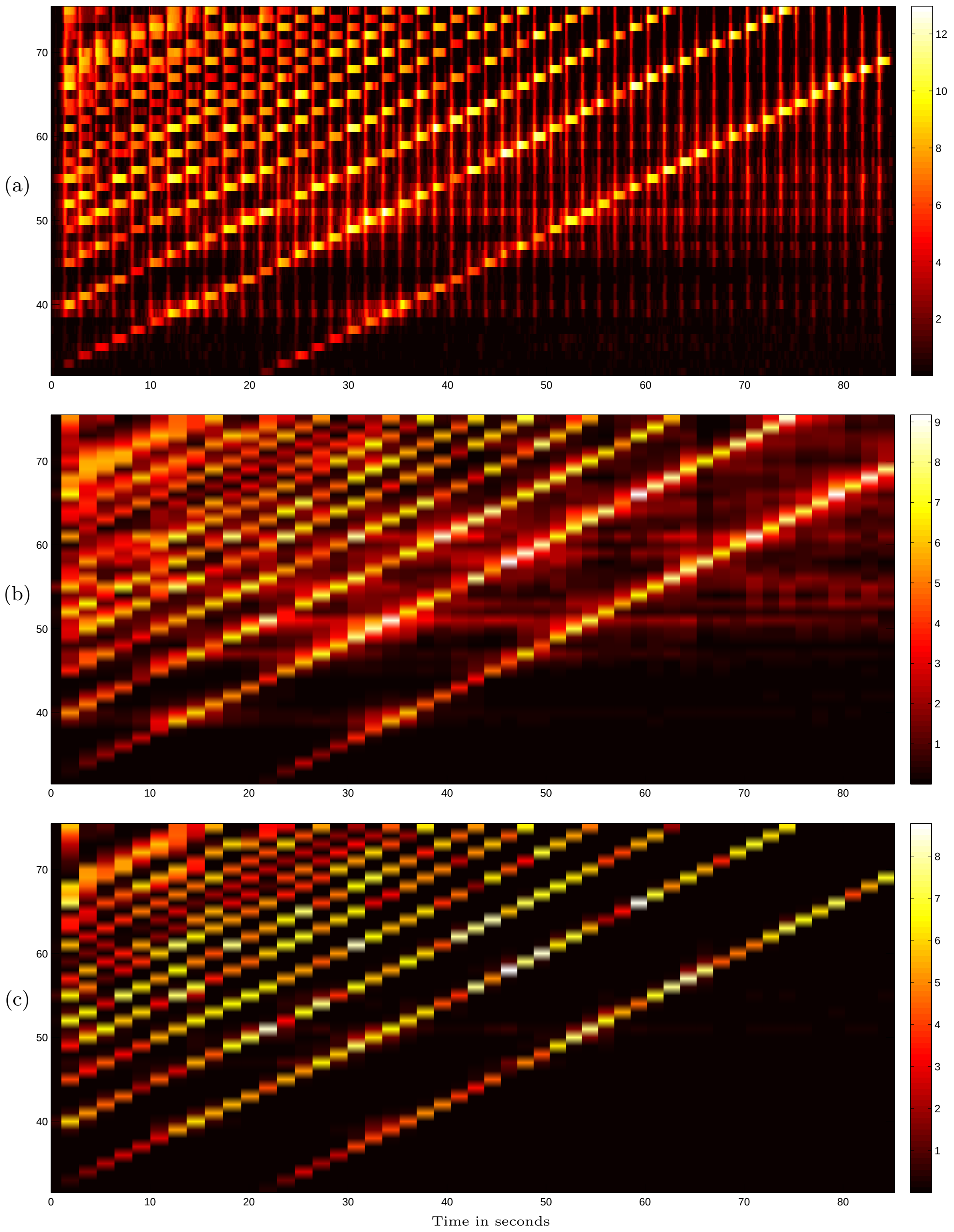


Figure 4.3: Various pitch representations of a chromatic scale played on a piano, showing the pitches $p \in [32 : 75]$. **(a)** Fixed-length segmentation with window length $w = 0.1$s. **(b)** Adaptive segmentation with $\alpha = 1.0$. **(c)** Adaptive segmentation with $\alpha = 0.5$.

5 Evaluation

While we have seen how time-aware features can be obtained from an audio signal, it remains yet to be determined whether adaptive segmentation can in fact improve the quality of the resulting feature representations. To answer this question, we conducted an extensive evaluation and present the results in this chapter.

The evaluation setup is detailed in Section 5.1, where we describe of the datasets and spectral features used for the evaluation as well as the various adaptive segmentations we constructed from rhythmical information. An application-independent analysis is presented in Section 5.2, where we employ an *entropy* measure that correlates to the presence of noise in the feature sequence. To establish an understanding for the effect of time-aware features on different music processing tasks, we compare the performance of two MIR algorithms that are given both time-aware and fixed-length feature sequences as inputs. We present results for a *chord recognition* task in Section 5.3 as well as an *audio matching* problem in Section 5.4.

5.1 Evaluation Setup

To conduct our experiments, we use a diverse collection of real-world audio data, corresponding to roughly 36.5 hours of audio in total. Our collection consists of five datasets with beat annotations, see Table 5.1 for an overview.

	Files [#]	Length [sec]	Beats [#]	Mean Tempo [BPM]	Std. Tempo [%]
BEATLES	179	28831	52729	116.7	3.3
MAZURKA	298	45177	85163	126.0	24.6
RWC-POP	100	24406	43659	111.7	1.1
RWC-JAZZ	50	13434	19021	89.7	4.5
RWC-CLASSIC	61	19741	32733	104.8	15.2

Table 5.1: The five beat-annotated datasets used in our experiments.

The BEATLES datasets consists of 179 Beatles songs and can be generally classified as a representative of the Pop music genre. The majority of the songs employs a standard instrumentation composed of guitars, bass and drums as well as vocals and is stable with respect to tempo and meter. We use the beat annotations created by Matthew Davies [7]. The MAZURKA datasets contains piano recordings of five Mazurkas composed by Frederik Chopin. Each Mazurka is interpreted by a multitude of different performers, yielding a total of 298 audio files. The recorded performances, which clearly belong to the Classical music genre, are very expressive and therefore exhibit significant differences in tempo, articulation and note execution. The remaining datasets RWC-POP, RWC-JAZZ and RWC-CLASSIC are taken from the RWC music database [15] and feature a representative selection of pieces from the respective genres. The recordings of each datasets have varying instrumentation and contain percussive as well as non-percussive passages, some with high rhythmic complexity.

Our experiments rely on the spectral feature representations we have introduced in Chapter 2. More precisely, we test the effect of adaptive segmentation with the help of the pitch representation (Pitch) as introduced in Section 2.2 as well as Chroma-Pitch (CP) and Chroma-Log-Pitch (CLP(C)) features with varying compression factors C, see Section 2.3. Furthermore, we use CENS features and CRP(55) features in our experiments, as described in Section 2.4 and Section 2.5 respectively.

The most attention in our evaluation is directed towards the segmentations we use to derive the aforementioned spectral feature representations. To obtain comparative values, we employ the standard fixed-length segmentation technique with varying window sizes and no overlap. Fixed-length segmentation is denoted by FS(w) with w being the window size in seconds, e.g. $w = 0.1$. To construct adaptive segmentations of the audio signals, we use the time positions of various metrical pulses described in Chapter 3 as segment boundaries. More precisely, we construct a *beat segmentation* of the signal, using the reference beat annotations from our datasets. The symbol BS(α) is used to denote a beat segmentation. To measure the effect of the α-parameter described in Section 4.3, we construct our adaptive segmentations considering two different settings $\alpha = 1.0$ and $\alpha = 0.5$. The beat segmentation consists of relatively large segments, typically around 0.5 seconds in length. A more fine-grained segmentation is obtained on the basis of the predominant local periodicity, described in Section 3.2. This segmentation, which we call *pulse segmentation* PS(α), is constructed using the peaks of a PLP curve as segment boundaries. The PLP curve that serves as the basis for the pulse segmentation uses an unconstrained set of tempo parameters, thus it always reflects the predominant local pulse and may freely switch between semantic pulse levels. Hence, the resulting segmentation exhibits large variations in segment length. The last adaptive segmentation we consider is the *constrained pulse segmentation* CPS(α, τ). Here, we constrain the set of tempo parameters used in the PLP computation to a tempo octave around a multiple τ of the annotated tempo. Setting $\tau = 1$ should yield a quarter note segmentation of the signal and thus approximate the beat segmentation without relying on human reference annotations. In our experiments, we use $\tau = 1$ and $\tau = 2$. The average segment lengths of all described segmentations are shown in Table 5.2.

	BEATLES	MAZURKA	RWC-CLASSIC	RWC-POP	RWC-JAZZ	Mean
FS(0.1)	0.100	0.100	0.100	0.100	0.100	0.100
FS(0.5)	0.500	0.500	0.500	0.500	0.500	0.500
FS(1.0)	1.000	1.000	1.000	1.000	1.000	1.000
BS(1.0)	0.514	0.477	0.572	0.537	0.669	0.553
BS(0.5)	0.257	0.238	0.286	0.268	0.334	0.276
PS(1.0)	0.227	0.237	0.237	0.201	0.202	0.220
PS(0.5)	0.113	0.118	0.118	0.100	0.101	0.110
CPS(1.0, 1)	0.367	0.383	0.543	0.423	0.522	0.447
CPS(0.5, 1)	0.183	0.191	0.271	0.211	0.261	0.223
CPS(1.0, 2)	0.245	0.239	0.295	0.259	0.287	0.265
CPS(0.5, 2)	0.122	0.119	0.147	0.129	0.143	0.132

Table 5.2: Mean segment lengths in seconds for all segmentations and datasets used in our evaluation.

5.2 Entropy-Based Evaluation

The first part of our evaluation is devoted to analyzing the effect of using adaptive segmentation for feature extraction in comparison to the classical fixed-length approach. In general, we expect feature representations that have been obtained from adaptive segmentations to be “cleaner” than their fixed-length counterparts in the sense that each individual feature vector carries information about only one musical event. Contrary, feature sequences obtained via fixed-length segmentation suffer from the problem that each feature vector may be influenced by two or more musical events, which results in a blurred feature representation. This phenomenon has been demonstrated in Figure 1.1, Figure 4.1 and Figure 4.3. The cleaning effect of adaptive segmentation should be amplified by the noise removal technique we have introduced in Section 4.3.

In our first experiment, we aim at measuring this cleaning effect in an application-independent fashion. We consider a spectral feature vector to be “clean”, if most of the spectral energy is concentrated in a few components, while all other components have energy values close to zero. This property holds for feature vectors that capture only one musical event, whereas feature vectors influenced by several events exhibit a rather uniform energy distribution among their components. Given a single feature vector, the goal is now to quantify to what degree the energy distribution in this vector deviates from the uniform distribution. To this end, we use the *Shannon entropy*, an established measure in the field of information theory, where it is used to quantify the uncertainty associated with a random variable [42]. In our case, the entropy measure is used as follows. Given a D-dimensional feature vector $x = (x_1, x_2, ..., x_D)$ with $x_i \in \mathbb{R}_{\geq 0}$ for $i \in [1 : D]$, we first normalize x with respect to the ℓ^1-norm. Thereupon, it holds that $x_i \in [0, 1] \subset \mathbb{R}$ and $\sum_{i=1}^{D} x_i = 1$, thus x can be interpreted as a discrete probability distribution. The *entropy*

$H(x)$ of the feature vector x is then defined as

$$H(x) = -\sum_{i=1}^{D} x_i \log_2(x_i) \ . \tag{5.1}$$

Since the range of the entropy function $H(x)$ varies for different D, we define the *normalized entropy* $\bar{H}(x)$ as

$$\bar{H}(x) = \frac{H(x)}{\log_2(D)} \ . \tag{5.2}$$

The interpretation of $\bar{H}(x)$ for a feature vector x is as follows. If the spectral energy within x is distributed uniformly among its components, then $H(x) = 1$. Contrary, if all the energy is concentrated in one component, then $H(x) = 0$. Thus, the entropy value of a "clean" feature vector is close to zero whereas the entropy value of an "unclean" feature vector is close to one.

Our entropy-based evaluation was conducted using all five datasets described in Section 5.1. The individual entropy values were averaged across all feature sequences computed from all audio files in the entire collection. The final results are presented in Table 5.3.

	Pitch	CP	CLP(1)	CENS
FS(0.1)	0.476	0.619	0.736	0.502
FS(0.5)	0.518	0.666	0.821	0.546
FS(1.0)	0.549	0.703	0.861	0.582
BS(1.0)	0.481	0.675	0.834	0.559
BS(0.5)	**0.449**	0.637	0.782	0.529
PS(1.0)	0.487	0.631	0.772	0.510
PS(0.5)	0.466	0.607	**0.727**	0.492
CPS(1.0, 1)	0.518	0.663	0.823	0.540
CPS(0.5, 1)	0.484	0.624	0.772	0.509
CPS(1.0, 2)	0.494	0.637	0.785	0.515
CPS(0.5, 2)	0.464	**0.606**	0.738	**0.491**

Table 5.3: Average normalized entropy values $\bar{H}(x)$ for various features and segmentations. The best results for each spectral feature are indicated in bold face type.

The different spectral features used in this experiment have substantial influence on the resulting entropy values, thus making them hardly comparable. The lowest and therefore best entropy results, e.g. $\bar{H}(x) = 0.449$, are produced by the Pitch feature. This is due to the fact that the pitch feature vectors consist of many components ($D = 120$) of which many have values close to zero, in particular the components that correspond to low pitches. Also the CENS feature yields quite small entropy values, e.g. $\bar{H}(x) = 0.491$, which is a result of the quantization function that is applied to the CENS feature vectors. Contrary, the entropy values produced by the CP and CLP(1) features are rather high.

This is caused by small spurious energy values present in the pitch representation that are amplified when summing up over the pitch bands in the chroma binning step. The logarithmic compression performed in the $\texttt{CLP}(1)$ computation further boosts these small values, which results in the highest and therefore worst entropy scores. For this reason, we did not include $\texttt{CLP}(C)$ features with higher compression factors C in this experiment.

Looking at the various segmentations compared in this experiment, we can see that the best entropy values are all produced by adaptive segmentations. We can also observe that the entropy measure is very sensitive to the segment length, which is reflected by values of the fixed-length segmentations $\texttt{FS}(\cdot)$. Here, the entropy values increase with increasing segment lengths. However, when we compare fixed-length segmentations with adaptive segmentations that have approximately the same segment length w, for example $\texttt{FS}(0.1)$ with $w = 0.1$s and $\texttt{PS}(0.5)$ with average $w = 0.11$s, we can see that the entropy values for the adaptive segmentations are consistently lower. This suggests that the feature sequences obtained from adaptive segmentations are indeed “cleaner” than their fixed-length counterparts.

5.3 Chord Recognition

The automatic extraction of chord labels from audio recordings, usually called *chord recognition*, constitutes one of the central problems in music information retrieval. The multitude of contributions, e.g. [3, 6, 14, 17, 22, 27, 31, 32, 43, 46] reflects the importance of this task. Therefore, the second part of our evaluation is devoted to examining the effect of adaptive segmentation on the chord recognition problem. Broadly speaking, most automatic chord recognition procedures first convert a given audio signal into a chroma-based feature representation and then apply pattern matching techniques to map the chroma features to chord labels. In our experiments, we employ two different chord recognition strategies that are based on template matching and on Hidden Markov Models.

We begin by summarizing the template matching strategy as described in [21]. The general idea is to compute a set $\mathcal{T} \subset \mathcal{X}$ of templates that correspond to the set of chord labels $\mathcal{L}$. Since this method uses a chroma-based feature representation of the input signal, these templates can be though of as prototype chroma feature vectors where each vector represents a specific chord. Furthermore, we define a *distance function* $d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ that measures the similarity between two chroma feature vectors. In our experiments, we use the cosine measure for this purpose. The chord recognition is then performed in a framewise fashion, that is each feature vector is assigned the chord label $L \in \mathcal{L}$ that minimizes the distance to the corresponding template vector $T_L \in \mathcal{T}$. More precisely, let $x \in \mathcal{X}$ be a given feature vector, then the corresponding chord label L_x is given by

$$L_x = \underset{L \in \mathcal{L}}{\operatorname{argmin}}\; d(T_L, x)\ . \tag{5.3}$$

In our evaluation, we use a template set $\mathcal{T}$ that consists of 24 templates corresponding to the 12 major and 12 minor chords.

Our chord recognition experiments were conducted on the basis of the BEATLES dataset, using the publicly available ground-truth chord annotations by Mauch et al. [26]. In the first step, we reduce these annotations to the 24 chord labels represented by our template set $\mathcal{T}$ by mapping augmented to major chords and diminished to minor chords. We then quantize and segment the chord annotations to obtain a reference label for each feature vector. The evaluation is then performed framewise using standard precision and recall measures by comparing the computed labels with the reference labels. However, since we compare different segmentations in this experiment, we need to fix a common segmentation for the evaluation step in order to make the results comparable. To this end, we use a fixed-length segmentation FS(0.1) with window length $w = 0.1$s as the reference segmentation. The computed label sequences are re-segmented to this reference segmentation by means of our segmentation algorithm described in Section 4.2 using a majority vote aggregation function.

	CP	CLP(1)	CLP(10)	CLP(100)	CLP(1000)	CENS	CRP(55)
FS(0.1)	0.443	0.488	0.523	0.533	0.524	0.324	0.509
FS(0.5)	0.495	0.573	0.584	0.572	0.547	0.386	0.545
FS(1.0)	0.505	0.580	0.579	0.560	0.527	**0.403**	0.536
BS(1.0)	**0.506**	0.586	**0.596**	**0.584**	0.559	0.398	**0.555**
BS(0.5)	0.482	0.549	0.578	0.578	**0.564**	0.363	0.551
PS(1.0)	0.467	0.526	0.553	0.553	0.538	0.351	0.527
PS(0.5)	0.451	0.498	0.534	0.546	0.539	0.331	0.522
CPS(1.0, 1)	**0.506**	**0.583**	0.595	**0.584**	0.557	0.396	0.552
CPS(0.5, 1)	0.478	0.543	0.572	0.574	0.561	0.358	0.547
CPS(1.0, 2)	0.472	0.534	0.559	0.557	0.540	0.356	0.530
CPS(0.5, 2)	0.454	0.505	0.540	0.551	0.544	0.333	0.527

Table 5.4: F-measure values for the chord recognition experiment using the template-based chord recognition strategy. The best scores for each spectral feature are indicated in bold face type.

The results for the chord recognition experiment using the template matching strategy are shown in Table 5.4. The presented numbers are F-measure values obtained by averaging over all songs in the BEATLES dataset. Looking at the various spectral features used in this experiment, we can see that the best result $F = 0.596$ is produced by the CLP(10) feature. Overall, all CLP(C) features with varying compression factors C perform quite well, taking the four top places in this experiment. They are followed by the CRP(55) ($F = 0.555$) and CP ($F = 0.506$) features. The CENS features are far behind with an F-measure value of 0.403.

Regarding the different segmentations used in this experiment, we can observe a clear trend in favor of the adaptive segmentation techniques. Except for CENS features, all of the top scores are produced by adaptive segmentations, among which the beat segmentation BS(1.0) and the constrained pulse segmentation CPS(1.0, 1) yield the best results. In fact, the F-measure values for BS(1.0) and CPS(1.0, 1) are almost identical, e.g. $F = 0.596$

and $F = 0.595$ for the CLP(10) feature. This indicates that the CPS(1.0, 1) segmentation very well approximates the annotation-based beat segmentation BS(1.0). Hence, we can conclude that the constrained PLP curves successfully capture the tactus pulse of the Beatles songs due to their stable tempo and the presence of percussive instruments. However, the overall improvement of the adaptive segmentations compared with the fixed-length segmentations is rather small. The largest increase can be observed for the CLP(1000) feature, where the F-measure value improved from $F = 0.547$ for FS(0.5) to $F = 0.564$ for BS(0.5). Among the fixed-length approaches, FS(0.5) and FS(1.0) produce the best results. Surprisingly, the chord recognition procedure does not benefit from setting $\alpha = 0.5$, which should have a de-noising effect as described in Section 4.3. Only the CLP(1000) feature profits from this parameter setting, whose scores improve slightly, yet consistently, in comparison to the segmentations using $\alpha = 1.0$.

Since the presented template-based chord recognizer is rather simple and only yields moderate overall results, we use a more sophisticated procedure based on Hidden Markov Models (HMMs) in the next experiment. The HMM-based approach, which was originally proposed by She and Ellis in [43], is conceptually state-of-the-art and today the most widely used chord labeling technique. In contrast to the template-based approach, the HMM chord recognizer also accounts for the temporal context of the chords in the classification stage, which can be considered as a kind of context-aware filtering of the predicted chord labels. In this approach, the hidden states of the HMM correspond to the 24 chord labels specified above. The observation probabilities are obtained by replacing the chord templates by chord models, that are specified by a multivariate Gaussian distribution in terms of a mean vector μ and a covariance matrix Σ. In our experiment, μ and Σ are learned from a subset of the labeled BEATLES dataset. The transition matrix, which encodes the likelihood of passing over from one chord to any other chord, is determined from labeled training data as well. The final chord label sequence that jointly maximizes observation and transition probabilities, is then obtained via Viterby decoding. We refer to [3, 6, 27, 32, 43, 46] for details and various implementations of the HMM-based chord recognition approach.

We present the results of our second chord recognition experiment using the HMM-based approach in Table 5.5. The indicated F-values have been computed the same way as in the previous experiment. At first glance, we can see that the overall results improved significantly (best result $F = 0.755$) compared with the template matching technique (best result $F = 0.596$). Looking at the spectral feature side, we observe that the ranking has changed, as the CRP(55) features now yield the best score ($F = 0.755$). The previously top ranked feature CLP(100) is now in the third place ($F = 0.744$), closely behind the CLP(1000) feature with $F = 0.745$. The worst scores are again produced by the CENS and CP features, with the CP now taking the last place with $F = 0.572$.

Regarding the different segmentations compared in this experiment, the trend in favor of the adaptive segmentations is continuing. Again, all top scores are produced by adaptive segmentations, however, the best overall F-measure is now given by the pulse segmentation PS(0.5) with $F = 0.755$ instead of the beat segmentation BS(1.0) as in the previous experiment. In general, we can observe that shorter segments yield better results for the

	CP	CLP(1)	CLP(10)	CLP(100)	CLP(1000)	CENS	CRP(55)
FS(0.1)	0.530	0.646	0.702	0.727	0.735	0.589	0.727
FS(0.5)	0.566	0.685	0.689	0.676	0.669	0.609	0.695
FS(1.0)	0.541	0.633	0.619	0.602	0.593	0.579	0.624
BS(1.0)	**0.572**	**0.716**	0.731	0.729	0.714	0.621	0.708
BS(0.5)	0.551	0.694	0.725	0.731	0.734	0.604	0.729
PS(1.0)	0.563	0.694	0.725	0.725	0.720	0.621	0.733
PS(0.5)	0.558	0.676	0.725	0.738	0.739	0.611	**0.755**
CPS(1.0, 1)	**0.572**	0.710	0.719	0.708	0.698	0.621	0.710
CPS(0.5, 1)	0.555	0.694	0.725	0.721	0.717	0.605	0.736
CPS(1.0, 2)	0.569	0.704	0.730	0.732	0.730	**0.623**	0.732
CPS(0.5, 2)	0.559	0.684	**0.732**	**0.744**	**0.745**	0.614	0.753

Table 5.5: F-measure values for the chord recognition experiment using the HMM-based chord recognition strategy. The best scores for each spectral feature are indicated in bold face type.

CLP(C) feature with $C \in \{10, 100, 1000\}$ and the CRP(55) feature, which is clearly reflected by the fixed-length segmentations. This observation is also supported by the F-measure values produced by the PS(0.5) and CPS(0.5, 2) segmentations, which exhibit the shortest segment lengths among the adaptive segmentations. Comparing the fixed-length and adaptive segmentation scores, we can see a slightly larger improvement than in the previous experiment, e.g. for the CLP(10) feature, whose F-measure value increased from $F = 0.702$ for FS(0.1) to $F = 0.732$ for CPS(0.5, 2).

5.4 Audio Matching

The identification and retrieval of semantically related music data is of major concern in the field of music information retrieval. One typical instance of this problem is *cover song identification*, where one tries to identify all performances of the same piece by different artists in face of differences in instrumentation, articulation and tempo [41]. For the last part of our evaluation, we use a similar yet more local scenario called *audio matching*. Here, the goal is to automatically retrieve all passages from a set of audio files that musically correspond to a given query excerpt. To this end, we use a matching procedure described in [29], which we summarize in the following.

Let Q be a query excerpt and $(D_1, D_2, ...D_N)$ a set of audio files. In the first step, all audio files $D_1, D_2, ...D_N$ are concatenated to form single large audio file D, which we call *database document*. The goal is now to identify all passages within D that are musically similar to Q. To this end, we transform the query and database document into suitable feature representations $X = (x_1, x_2, ...x_K)$ and $Y = (y_1, y_2, ..., y_L)$ with $x, y \in \mathcal{X}$. We then define a *cost measure* $c : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ to quantify the (dis-)similarity between two feature vectors from $\mathcal{X}$. In our experiments, we simply use the cosine measure for this purpose. To identify the relevant passages in the database document, we employ a distance function Δ that compares the query feature sequence X with subsequences of the database feature sequence Y. To cope with possible variations in tempo, the distance function is defined using *dynamic time warping* (DTW) that enables the matching procedure to accurately align subsequences of different lengths. More precisely, the distance function $\Delta : [1 : L] \rightarrow \mathbb{R} \cup \{\infty\}$ between X and Y is given by

$$\Delta(l) = \frac{1}{K} \min_{a \in [1:l]} (\mathrm{DTW}(X, Y(a : l))) \tag{5.4}$$

where $\mathrm{DTW}(X, Y(a : l))$ denotes the DTW distance between X and the subsequence $Y(a : l)$ with respect to the cost measure c. For details regarding DTW and the distance function, we refer to [28].

The distance function Δ can be interpreted as follows. A small value of $\Delta(l)$ for some $l \in [1 : L]$ indicates that the subsequence $Y(a_l, l)$ starting at a_l and ending in l is similar to the query X. Conversely, $\Delta(l)$ attains a large value if the subsequence ending in l bears no resemblance to X. To determine the best match within Y, one simply identifies the index $l_0 \in [1 : L]$ that minimizes Δ, yielding the subsequence $Y(a_{l_0} : l_0)$. The value $\Delta(l_0)$ is also referred to as *cost* of the match. To find the second best match, we exclude a neighborhood around l_0 from further consideration to avoid large overlaps with the best match. In our experiments, we use half the query length to the left and right of l_0 to define such a neighborhood. The Δ-values within this neighborhood are excluded by setting them to ∞. Subsequent matches can then be obtained by repeating the above procedure.

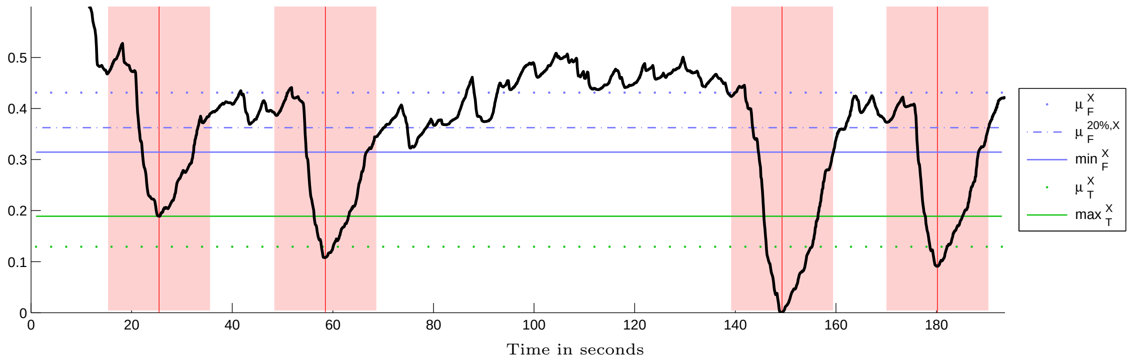


Figure 5.1: Distance function Δ with respect to the query E_3 using the Shostakovich Waltz as database document. The semantically correct matches E_1, E_2, E_3 and E_4 are indicated by vertical red lines. The excluded neighborhoods are shown in light red, the false alarm region consists of all indices outside these areas. The various statistics underlying the quality measures are indicated by the horizontal lines.

We illustrate the behavior of the distance function Δ in Figure 5.1 by means of our Shostakovich example. Here, we have transformed the query and database audio signals into CRP feature representations. Using the passage E_3 (trombone) as query excerpt, the resulting distance function Δ clearly reveals the ending positions of the passages E_1, E_2,

E_3 and E_4 in form of four significant local minima. The light red regions around the four matches denote the excluded neighborhoods.

In order to correctly identify relevant matches using the described procedure, it is important that Δ fulfills two conditions. First, the local minima of Δ that correspond to true matches should be close to zero to avoid false negatives. Second, in order to prevent false positives, Δ should attain values well above zero in the regions outside of the excluded neighborhoods, which we refer to as *false alarm region*. In accordance with [29], we now introduce several quality measures that aim to capture the degree of compliance to these conditions for a given Δ. We begin by defining μ_T^X and $\max_T^X$ as the average respectively maximum value of Δ over all indices that correspond to the true matches for a given query X. Analogously, μ_F^X and $\min_F^X$ denote the average respectively minimum value of Δ considering all indices within the false alarm region. These values are indicated in Figure 5.1 by horizontal lines. In order to distinguish between true matches and spurious ones, μ_T^X and $\max_T^X$ should be small while μ_F^X and $\min_F^X$ should be large. The quality measures α^X and γ^X express these properties as a single number and are defined as the quotients $\alpha^X = \mu_T^X / \mu_F^X$ and $\gamma^X = \max_T^X / \min_F^X$.

The interpretation of these quality measures is as follows. Small values for α^X and γ^X indicate, that Δ allows for good separability of true and spurious matches. If $\gamma^X < 1$ then all true matches appear as the top most matches whereas $\gamma^X > 1$ means that at least one false positive match appears before all true matches are retrieved. The quality measure γ^X is quite strict as one single outlier (either a true match with high cost or a spurious match with low cost in the false alarm region) may completely degenerate the value of γ^X. In contrast, the quality measure α^X is quite lenient in the sense that it may still attain a low value even if a large number of false positive matches is retrieved. As a tradeoff between α^X and γ^X, we introduce a third quality measure β^X. To this end, we sort the indices within the false alarm region by increasing cost and define $\mu_F^{p\%,X}$ as the average value of Δ over the lower $p\%$ of these indices. The quality measure β^X is then defined as $\beta^X = \mu_T^X / \mu_F^{p\%,X}$. In our experiments, we use $p = 20$, thus only considering the lower 20% of the indices within the false alarm region. Note that β^X is more robust to outliers than γ^X while being more sensitive to false positive matches than α^X.

In the following, we present the results of the experiments we conducted to evaluate the effect of adaptive segmentation for the audio matching task. More precisely, the main objective of the experiment is to assess the discriminative power of various time-aware features in comparison to fixed-length features. The experiments were performed using the `MAZURKA` dataset, which consists of five Mazurkas in many different interpretations. Therefore, it offers a multitude of semantically related music material, which is perfectly suited for the audio matching scenario. We carefully selected five query excerpts from each of the Mazurkas with an average length of 20 seconds, yielding a total of 25 queries. For each query X, we computed the quality measures α^X, β^X and γ^X for various features and segmentations using the entire `MAZURKA` dataset as database documents. By averaging the over all 25 queries, we obtain the numbers for α, β and γ that are shown in Table 5.6.

Looking at the spectral features used in this experiment, we can see that `CRP`(55) fea-

	CLP(100)			CENS			CRP(55)		
	α	β	γ	α	β	γ	α	β	γ
FS(0.1)	0.278	0.363	**0.929**	0.305	**0.390**	**0.941**	0.236	0.309	**0.772**
FS(0.5)	0.296	0.408	1.209	0.371	0.480	1.190	0.237	0.322	0.862
FS(1.0)	0.348	0.489	1.473	0.468	0.611	1.437	0.276	0.384	1.020
BS(1.0)	0.265	0.382	1.786	0.354	0.472	1.932	**0.216**	**0.300**	1.316
BS(0.5)	**0.257**	**0.361**	1.588	**0.299**	0.402	1.546	0.225	0.309	1.095
PS(1.0)	0.356	0.467	1.691	0.396	0.504	1.634	0.304	0.397	1.568
PS(0.5)	0.358	0.463	1.632	0.376	0.477	1.587	0.320	0.414	1.563
CPS(1.0, 1)	0.294	0.398	1.115	0.374	0.481	1.178	0.235	0.322	0.798
CPS(0.5, 1)	0.284	0.383	1.033	0.328	0.426	1.071	0.240	0.325	0.820
CPS(1.0, 2)	0.279	0.372	1.028	0.329	0.424	1.091	0.228	0.307	0.798
CPS(0.5, 2)	0.282	0.371	0.942	0.308	0.397	0.984	0.243	0.321	0.799

Table 5.6: Results of the audio matching evaluation for various features and segmentations. The best scores for each quality measure are indicated in bold face type.

tures clearly yield the best scores for all quality measures, e.g. $\alpha = 0.216$. We conjecture that this is due to the fact that CRP(55) features have been designed and optimized for matching and retrieval scenarios and therefore exhibit the best discriminative power. The rather general-purpose CLP(100) features hold the second place among the spectral features, closely followed by the CENS features.

Unfortunately, the results of this experiment do not allow for a clear statement regarding the benefit of adaptive segmentation for audio matching. Especially in terms of the quality measure γ, the classical fixed-length segmentation approach FS(0.1) with window length $w = 0.1$s outperforms all other segmentation techniques. Even though the beat segmentations BS(1.0) and BS(0.5) yield the best scores for α and β for CLP(100) and CRP(55) features, the improvement over FS(0.1) is rather small. Furthermore, BS(1.0) produces by far the worst γ values. Also far behind with respect to all of the quality measures are the unconstrained pulse segmentations PS(1.0) and PS(0.5). This results from the fact that the recorded Mazurka performances are very expressive and therefore exhibit large deviations in local tempo and articulation. For this reason, the PLP curves used to construct the pulse segmentations fail to capture the strongly distorted local pulse. This causes the various Mazurka versions to be segmented very differently, which makes it hard, if not impossible, for the DTW algorithm to correctly align semantically equivalent passages. This problem is mitigated by constraining the tempo parameters of the PLP curve which is reflected in the scores of the various constrained pulse segmentations CPS$(\cdot, \cdot)$. Their γ values are even quite close to the scores produced by FS(0.1). Finally, we observe that the α and β values of the CRP(55) feature are better for BS(1.0) ($\alpha = 0.216$) than for BS(0.5) ($\alpha = 0.225$). This suggests that CRP(55) features already have a de-noising effect and thus do not benefit from the noise removal technique described in Section 4.3.

In view of the strong temporal deviations between different recordings of the same

Mazurka, we slightly modify the matching procedure for the last experiment of our evaluation. So far, the distance function Δ has been computed using DTW to align the query feature sequence to corresponding subsequences in the database document. The use of DTW is necessary because two musically equivalent feature sequences can differ in length due to the variations in tempo introduced by the different playing styles of the performers. However, assuming that the rhythmical analysis we perform to construct the adaptive segmentations correctly reveals the time positions of the musical events that make up the piece, all musically equivalent time-aware feature sequences should have equal length. In other words, a rhythmically correct adaptive segmentation compensates for the temporal deviations between two different interpretations of the same piece. This property makes the DTW step in the audio matching procedure obsolete. We therefore disable DTW in this last experiment, thereby forcing the matching procedure to align the feature sequences in a one-to-one fashion. The results are shown in Table 5.7.

	CLP(100)			CENS			CRP(55)		
	α	β	γ	α	β	γ	α	β	γ
FS(0.1)	0.676	0.806	1.779	0.675	0.777	1.623	0.643	0.735	1.583
FS(0.5)	0.615	0.789	2.085	0.644	0.776	1.708	0.573	0.683	1.732
FS(1.0)	0.569	0.764	2.221	0.644	0.800	1.809	0.523	0.654	1.740
BS(1.0)	0.196	0.253	0.738	0.269	0.329	0.819	**0.155**	**0.185**	0.504
BS(0.5)	**0.188**	**0.234**	**0.666**	**0.222**	**0.266**	**0.721**	0.161	0.189	**0.495**
PS(1.0)	0.674	0.823	1.880	0.687	0.804	1.660	0.648	0.751	1.630
PS(0.5)	0.693	0.823	1.745	0.692	0.797	1.605	0.670	0.768	1.570
CPS(1.0, 1)	0.525	0.655	1.732	0.558	0.667	1.550	0.477	0.570	1.538
CPS(0.5, 1)	0.545	0.664	1.673	0.558	0.656	1.540	0.510	0.598	1.527
CPS(1.0, 2)	0.560	0.678	1.607	0.574	0.671	1.464	0.519	0.603	1.457
CPS(0.5, 2)	0.585	0.693	1.545	0.583	0.673	1.425	0.550	0.631	1.424

Table 5.7: Results of the audio matching evaluation without using DTW for various features and segmentations. The best scores for each quality measure are indicated in bold face type.

In comparison to the previous experiment, there is no change regarding the performance of the different spectral features, CRP(55) still yields the best values for all of the quality measures. However, the scores produced by the beat segmentations BS(1.0) and BS(0.5) drastically improve, compare e.g. $\gamma = 0.666$ versus $\gamma = 1.588$ for CLP(100). In fact, not only do they clearly outperform all other segmentation techniques but also yield significantly better scores than any approach achieved in the previous experiment. This indicates that DTW effectively degrades the performance of the beat segmentation for the audio matching task. The scores produced by the fixed-length segmentations show a clear decline, which is not surprising as they rely on DTW for the alignment of the feature sequences. Also, the performance of the pulse segmentations is rather poor, for the same reasons as in the previous experiment. Finally, it should be noted that this comparison is somewhat unfair as the beat segmentation was constructed using human reference annotations. Yet, this experiment conclusively demonstrates the power of adaptive segmentation.

6
Summary

In this thesis, we presented an in-depth analysis of various segmentation techniques for musical audio signals for the purpose of feature extraction. Our main focus was directed towards adaptive segmentation, which makes use of rhythmical information to define musically meaningful segment boundaries. We compared this approach with traditional fixed-length segmentation procedures that rely on predefined parameters to partition the audio signal.

To this end, we first reduced the complex music phenomenon to its two main dimensions, which were then separately examined. The spectral dimension was explored in Chapter 2, where we introduced a variety of feature representations that characterize a music signal by its spectral content. Starting with a simple pitch decomposition of the audio signal, we successively refined this approach to obtain representations that capture relevant harmonic aspects while being robust to noise and variations in instrumentation and timbre.

Chapter 3 was devoted to the temporal dimension of music. Here, we gave an overview of several techniques for the analysis of the rhythmical structure of musical audio signals. In particular, we focused on the extraction of the periodic pulses that constitute the basis of rhythm.

In Chapter 4, both musical dimensions were combined in the adaptive segmentation procedure. We described how to construct rhythmically meaningful segmentations of the music signal which then served as the basis for computing spectral feature representations. Furthermore, we presented a flexible and extensible segmentation algorithm and introduced a method to remove noisy parts from the audio signal.

Finally, we conducted a comprehensive evaluation of our adaptive segmentation technique, which was presented in Chapter 5. Using a multitude of spectral features, we compared adaptive segmentations constructed from different metrical pulses and fixed-length segmentations with varying window lengths. We found that the rhythmical information used to construct the adaptive segmentations significantly influences the resulting feature sequences. Furthermore, we observed that the time-aware feature representations generally exhibit sharper feature differences than their fixed-length counterparts and show a high degree of tempo-invariance. Based on these results, we conclude that adaptive segmentation is indeed well-suited for segmenting musical audio signals and is clearly superior to the traditional fixed-length approach.

A

Source Code

In this chapter, the headers of selected MATLAB functions created during the writing of this thesis are reproduced. The headers contain information about the name of the described function and its input/output behavior.

Segmentation Algorithm

The `resample_adaptive` function implements the segmentation algorithm we described in Section 4.2.

Sample usage:
`[out,T_out] = resample_adaptive(f_chroma, 50, beat_segmentation);`

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Name: resample_adaptive
% Version: 1.0
% Date: 21.03.2011
% Programmer: Philipp von Styp-Rekowsky
%
% Description:
%   Segments a feature sequence according to a given adaptive
%   segmentation.
%
% Input:
% - feature: Feature sequence to be segmented, specified as n-by-d matrix
%   with time progressing within rows.
% - T_in: Either a scalar denoting the feature rate of the feature sequence
%   or a segmentation. A segmentation is specified as a n-by-2 matrix where
%   each row corresponds to a segment with the start time [sec] given in
%   the first column and the end time [sec] in the 2. column. Must be
%   sorted increasing order of the start times.
% - T_out: Either a scalar denoting the desired feature rate or the desired
%   segmentation.
% - parameter
%            .alpha: Ratio of output segment length to consider in the
%             segmentation process (0 <= alpha <= 1; Default: 1)
%            .aggregation_strategy: Strategy used to reduce a feature
%             sequence to a single vector (Possible values: 'sum',
%             'weighted_sum', 'majority_vote'; Default: 'weighted_sum')
%            .default_sample: Default value to assume for possible gaps in
%             the feature sequence (Default: 0)
%            .window_func: Function handle for an additional window for the
%             aggregation step. The function must accept a single
%             scalar specifying the desired window size (in samples). Only
%             active in combination with the aggregation strategies 'sum'
%             and 'weighted_sum'. (Default: @rectwin)
%
% Output:
% - out: Segmented feature sequence
% - T_out: Resulting segmentation of the feature sequence
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Audio Player

We implemented a simple audio player since MATLAB lacks any reasonable audio playback functionality. It offers a simple GUI to play, pause and stop the playback as well as to jump to any desired time position in the signal. The signal can either be passed as a variable or loaded from a specified file. The functionality of the player can be extended by suitable plugins. Use the player function to start the audio player.

Sample usage:

```
player();

% Load audio signal from MATLAB workspace
player(f_audio,fs);

% Load audio file
player(filename);
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Name: player
% Version: 1.0
% Date: 21.03.2011
% Programmer: Philipp von Styp-Rekowsky
%
% Description:
%   Starts the Audio Player.
%
% Input:
%   The player accepts the following types of inputs:
%   - None
%   - Audio signal:
%     - f_audio: Mono or stereo audio signal specified as n-by-m matrix
%       with time progressing within rows.
%     - fs: Sampling rate (Optional; Default: 22050)
%   - File name:
%     - filename: Path to an audio file
%
% Output:
% - handle: Handle to the audio player instance
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Bibliography

[1] M. A. Bartsch and G. H. Wakefield, *To catch a chorus: Using chroma-based representations for audio thumbnailing*, in Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA), New Paltz, NY, USA, 2001, pp. 15–18.

[2] J. P. Bello, L. Daudet, S. Abdallah, C. Duxbury, M. Davies, and M. B. Sandler, *A tutorial on onset detection in music signals*, IEEE Transactions on Speech and Audio Processing, 13 (2005), pp. 1035–1047.

[3] J. P. Bello and J. Pickens, *A robust mid-level representation for harmonic content in music signals*, in Proceedings of the International Conference on Music Information Retrieval (ISMIR), London, UK, 2005.

[4] J. Bilmes, *A model for musical rhythm*, in Proceedings of the International Computer Music Conference (ICMC), San Jose, USA, 1992.

[5] H.-T. Cheng, Y.-H. Yang, Y.-C. Lin, and H. H. Chen, *Multimodal structure segmentation and analysis of music using audio and textual information*, in Proceedings of the IEEE International Symposium on Circuits and Systems, Taipei, Taiwan, 2009, pp. 1677–1680.

[6] T. Cho, R. J. Weiss, and J. P. Bello, *Exploring common variations in state of the art chord recognition systems*, in Proceedings of the Sound and Music Computing Conference (SMC), Barcelona, Spain, 2010, pp. 1–8.

[7] M. E. P. Davies, N. Degara, Mark, D. Plumbley, M. E. P. Davies, N. Degara, and M. D. Plumbley, *Evaluation methods for musical audio beat tracking algorithms*, 2009.

[8] S. B. Davis and P. Mermelstein, *Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences*, Readings in Speech Recognition, (1990), pp. 65–74.

[9] D. Ellis and G. Poliner, *Identifying cover songs with chroma features and dynamic programming beat tracking*, in Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), Honolulu, Hawaii, USA, 2007.

[10] D. P. W. Ellis, *Beat tracking by dynamic programming*, Journal of New Music Research, 36 (2007), pp. 51–60.

[11] D. P. W. Ellis, C. V. Cotton, and M. I. Mandel, *Cross-correlation of beat-synchronous representations for music similarity*, in Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), Taipei, Taiwan, 2008, pp. 57–60.

[12] A. J. Eronen and A. P. Klapuri, *Musical instrument recognition using cepstral coefficients and temporal features*, in Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), vol. 2, Istanbul, Turkey, 2000, pp. II753–II756.

[13] ——, *Music tempo estimation with k-nn regression*, IEEE Transactions on Audio, Speech, and Language Processing, 18 (2010), pp. 50–57.

[14] T. Fujishima, *Realtime chord recognition of musical sound: A system using common lisp music*, in Proc. ICMC, Beijing, 1999, pp. 464–467.

[15] M. Goto, H. Hashiguchi, T. Nishimura, and R. Oka, *RWC music database: Popular, classical and jazz music databases*, in Proceedings of the International Conference on Music Information Retrieval (ISMIR), Paris, France, 2002.

[16] P. Grosche and M. Müller, *Computing predominant local periodicity information in music recordings*, in Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA), New Paltz, New York, USA, 2009.

[17] C. Harte and M. Sandler, *Automatic chord identification using a quantised chromagram*, in Proceedings of the 118th AES Convention, Barcelona, Spain, 2005.

[18] C. Harte, M. Sandler, and M. Gasser, *Detecting harmonic change in musical audio*, in Proceedings of the ACM Workshop on Audio and Music Computing Multimedia, Santa Barbara, California, USA, 2006, pp. 21–26.

[19] J. H. Jensen, M. G. Christensen, D. P. W. Ellis, and S. H. Jensen, *A tempo-insensitive distance measure for cover song identification based on chroma features*, in Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), Las Vegas, USA, 2008.

[20] R. D. John, H. L. John, and G. P. John, *Discrete-time processing of speech signals*, IEEE Press, (1999).

[21] V. Konz, M. Müller, and S. Ewert, *A multi-perspective evaluation framework for chord recognition*, in Proceedings of the 11th International Conference on Music Information Retrieval (ISMIR), Utrecht, Netherlands, 2010, pp. 9–14.

[22] K. Lee and M. Slaney, *Acoustic chord transcription and key extraction from audio using key-dependent hmms trained on synthesized audio*, IEEE Transactions on Audio, Speech, and Language Processing, 16 (2008), pp. 291–301.

[23] M. Levy and M. Sandler, *Structural segmentation of musical audio by constrained clustering*, IEEE Transactions on Audio, Speech and Language Processing, 16 (2008), pp. 318–326.

[24] N. C. Maddage, M. Kankanhalli, and H. Li, *Effectiveness of signal segmentation for music content representation*, in Advances in Multimedia Modeling, S. Satoh, F. Nack, and M. Etoh, eds., vol. 4903 of Lecture Notes in Computer Science, Springer Berlin / Heidelberg, 2008, pp. 477–486.

[25] N. C. Maddage, C. Xu, M. S. Kankanhalli, and X. Shao, *Content-based music structure analysis with applications to music semantics understanding*, in Proceedings of the ACM International Conference on Multimedia, New York, NY, USA, 2004, pp. 112–119.

[26] M. Mauch, C. Cannam, M. Davies, S. Dixon, C. Harte, S. Kolozali, D. Tidhar, and M. Sandler, *OMRAS2 metadata project 2009*, in Proceedings of the International Conference on Music Information Retrieval (ISMIR), Kobe, Japan, 2009.

[27] M. Mauch and S. Dixon, *Simultaneous estimation of chords and musical context from audio*, IEEE Transactions on Audio, Speech, and Language Processing, 18 (2010), pp. 1280–1289.

[28] M. Müller, *Information Retrieval for Music and Motion*, Springer, 2007.

[29] M. Müller and S. Ewert, *Towards timbre-invariant audio features for harmony-based music*, vol. 18, 2010, pp. 649–662.

[30] M. Müller, F. Kurth, and M. Clausen, *Audio matching via chroma-based statistical features*, in Proceedings of the 6th International Conference on Music Information Retrieval (ISMIR), 2005, pp. 288–295.

[31] L. Oudre, Y. Grenier, and C. Févotte, *Template-based chord recognition: Influence of the chord types*, in Proceedings of the 10th International Conference on Music Information Retrieval (ISMIR), Kobe, Japan, 2009.

[32] H. Papadopoulos and G. Peeters, *Large-scale study of chord estimation algorithms based on chroma representation and HMM*, in Content-Based Multimedia Indexing (CBMI), 2007, pp. 53–60.

[33] ——, *Joint estimation of chords and downbeats from an audio signal*, IEEE Transactions on Audio, Speech, and Language Processing, 19 (2011), pp. 138–152.

[34] J. Paulus and A. Klapuri, *Music structure analysis using a probabilistic fitness measure and a greedy search algorithm*, IEEE Transactions on Audio, Speech, and Language Processing, 17 (2009), pp. 1159–1170.

[35] J. Paulus, M. Müller, and A. Klapuri, *Audio-based music structure analysis*, in Proceedings of the 11th International Conference on Music Information Retrieval (ISMIR), Utrecht, Netherlands, 2010, pp. 625–636.

[36] S.-C. Pei and N.-T. Hsu, *Instrumentation analysis and identification of polyphonic music using beat-synchronous feature integration and fuzzy clustering*, Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), (2009), pp. 169–172.

[37] L. Rabiner and B.-H. Juang, *Fundamentals of Speech Recognition*, Prentice Hall Signal Processing Series, 1993.

[38] C. S. Sapp, *Comparative analysis of multiple musical performances*, in Proceedings of the International Conference on Music Information Retrieval (ISMIR), Vienna, Austria, 2007, pp. 497–500.

[39] E. D. Scheirer, *Tempo and beat analysis of acoustical musical signals*, Journal of the Acoustical Society of America, 103 (1998), pp. 588–601.

[40] J. Seppänen, *Tatum grid analysis of musical signals*, in Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA), 2001, pp. 131–134.

[41] J. Serrà, E. Gómez, P. Herrera, and X. Serra, *Chroma binary similarity and local alignment applied to cover song identification*, IEEE Transactions on Audio, Speech and Language Processing, 16 (2008), pp. 1138–1151.

[42] C. E. Shannon, *A mathematical theory of communication*, Bell System Technical Journal, 27 (1948), pp. 379–423.

[43] A. Sheh and D. P. W. Ellis, *Chord segmentation and recognition using EM-trained hidden Markov models*, in Proceedings of the International Conference on Music Information Retrieval (ISMIR), Baltimore, USA, 2003.

[44] A. M. Stark, M. E. P. Davies, and M. D. Plumbley, *Real-time beat-synchronous analysis of musical audio*, in Proceedings of the International Conference on Digital Audio Effects (DAFx), Como, Italy, 2009.

[45] G. Tzanetakis and P. Cook, *Musical genre classification of audio signals*, IEEE Transactions on Speech and Audio Processing, 10 (2002), pp. 293–302.

[46] Y. Ueda, Y. Uchiyama, T. Nishimoto, N. Ono, and S. Sagayama, *HMM-based approach for automatic chord detection using refined acoustic features*, in Proceedings of the 35nd IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), Dallas, USA, 2010.

[47] W. You and R. Dannenberg, *Polyphonic music note onset detection using semi-supervised learning*, in Proceedings of the International Conference on Music Information Retrieval (ISMIR), Vienna, Austria, 2007.

[48] R. Zhou, M. Mattavelli, and G. Zoia, *Music onset detection based on resonator time frequency image*, IEEE Transactions on Audio, Speech, and Language Processing, 16 (2008), pp. 1685–1695.

[49] E. Zwicker and H. Fastl, *Psychoacoustics, facts and models*, Springer Verlag, New York, NY, US, 1990.